VỤ GIÁO DỤC CHUYÊN NGHIỆP

GIÁO TRÌNH
CƠ SỞ DỮ LIỆU

SÁCH DÙNG CHO CÁC TRƯỜNG ĐÀO TẠO HỆ TRUNG HỌC CHUYÊN NGHIỆP





NHÀ XUẤT BẢN GIÁO DỤC

TÔ VĂN NAM

GIÁO TRÌNH
CƠ SỞ DỮ LIỆU

(Sách dùng cho các trường đào tạo hệ Trung học chuyên nghiệp)

(Tái bản lần thứ nhất)

NHÀ XUẤT BẢN GIÁO DỤC

Lời giới thiệu

Năm 2002, Vụ Giáo dục Chuyên nghiệp – Bộ Giáo dục và Đào tạo đã phối hợp với Nhà xuất bản Giáo dục xuất bản 21 giáo trình phục vụ cho đào tạo hệ THCN. Các giáo trình trên đã được nhiều trường sử dụng và hoan nghênh. Để tiếp tục bổ sung nguồn giáo trình đang còn thiếu, Vụ Giáo dục Chuyên nghiệp phối hợp cùng Nhà xuất bản Giáo dục tiếp tục biên soạn một số giáo trình, sách tham khảo phục vụ cho đào tạo ở các ngành : Điện – Điện tử, Tin học, Khai thác cơ khí. Những giáo trình này trước khi biên soạn, Vụ Giáo dục Chuyên nghiệp đã gửi đề cương về trên 20 trường và tổ chức hội thảo, lấy ý kiến đóng góp về nội dung đề cương các giáo trình nói trên. Trên cơ sở nghiên cứu ý kiến đóng góp của các trường, nhóm tác giả đã điều chỉnh nội dung các giáo trình cho phù hợp với yêu cầu thực tiễn hơn.

Với kinh nghiệm giảng dạy, kiến thức tích luỹ qua nhiều năm, các tác giả đã cố gắng để những nội dung được trình bày là những kiến thức cơ bản nhất nhưng vẫn cập nhật được với những tiến bộ của khoa học kỹ thuật, với thực tế sản xuất. Nội dung của giáo trình còn tạo sự liên thông từ Dạy nghề lên THCN.

Các giáo trình được biên soạn theo hướng mở, kiến thức rộng và cố gắng chỉ ra tính ứng dụng của nội dung được trình bày. Trên cơ sở đó tạo điều kiện để các trường sử dụng một cách phù hợp với điều kiện cơ sở vật chất phục vụ thực hành, thực tập và đặc điểm của các ngành, chuyên ngành đào tạo.

Để việc đổi mới phương pháp dạy và học theo chỉ đạo của Bộ Giáo dục và Đào tạo nhằm nâng cao chất lượng dạy và học, các trường cần trang bị đủ sách cho thư viện và tạo điều kiện để giáo viên và học sinh có đủ sách theo ngành đào tạo. Những giáo trình này cũng là tài liệu tham khảo tốt cho học sinh đã tốt nghiệp cần đào tạo lại, nhân viên kỹ thuật đang trực tiếp sản xuất.

Các giáo trình đã xuất bản không thể tránh khỏi những sai sót. Rất mong các thầy, cô giáo, bạn đọc góp ý để lần xuất bản sau được tốt hơn. Mọi góp ý xin gửi về : Công ty Cổ phần sách Đại học – Dạy nghề, 25 Hàn Thuyên – Hà Nội.

VỤ GIÁO DỤC CHUYÊN NGHIỆP - NXB GIÁO DỤC

Mở đầu

***Cơ sở dữ liệu** (CSDL) **là một trong những bộ môn không thể thiếu được trong chương trình của bất cứ môi trường đào tạo chuyên nghiệp nào, bởi vì có tới hơn 80% các ứng dụng của tin học là phục vụ công tác quản lý, mà **quản lý** thực chất là **quản lý thông tin**. Song, mọi thông tin cần quản lý trên máy tính cũng đều phải được thể hiện bằng các **dữ liệu**, do vậy khi nói đến **quản lý thông tin** tức là nói đến **quản lý dữ liệu**.*

*Cuốn "**Cơ sở dữ liệu**" bao gồm hai phần:*

Phần I : dành phần lớn nội dung giới thiệu các cấu trúc và khái niệm cơ bản về Cơ sở dữ liệu.

Phần II : cung cấp một số hướng dẫn đầy đủ cho bạn đọc về Hệ Quản trị Cơ sở dữ liệu SQL Server 2000.

Cuốn sách được biên soạn nhằm phục vụ công tác đào tạo và học tập trong các trường Trung học chuyên nghiệp và Cao đẳng, nhưng cũng là một tư liệu tham khảo tốt cho tất cả những ai quan tâm đến CSDL.

Tác giả rất hoan nghênh và cảm ơn mọi ý kiến đóng góp của các bạn đọc ở khắp mọi nơi. Thư góp ý xin gửi về theo địa chỉ:

TÔ VĂN NAM, Bộ môn Hệ thống thông tin, Khoa Công nghệ Thông tin - Trường Đại Học Bách Khoa, Hà Nội.

Email: namtv@it-hut.edu.vn

TÁC GIẢ

Phần 1. CẤU TRÚC VÀ KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU

Chương 1

CÁC KHÁI NIỆM CƠ BẢN

I. CÁC KHÁI NIỆM VỀ QUAN HỆ

1.1. Tập hợp

1.1.1. Các khái niệm về tập hợp

- Tập hợp là khái niệm đầu tiên của toán học, không định nghĩa. Ví dụ: Tập hợp các số nguyên tự nhiên; tập hợp các sinh viên trong một lớp; tập hợp các nghiệm của phương trình, ...
- Một tập hợp được tạo thành từ những phần tử của nó. Phần tử cũng là khái niệm đầu tiên của toán học, không định nghĩa.
- Người ta thường dùng các chữ cái viết hoa để đặt tên cho một tập hợp, chữ cái viết thường để chỉ một phần tử của tập hợp.

Ví dụ: N là tập hợp các số nguyên tự nhiên; X là tập hợp các sinh viên lớp 97 Đ1A, ...

- x góp phần tạo nên tập hợp A, ta nói x là một phần tử của A và ký hiệu là $x \in A$, đọc là: x thuộc A; y không là phần tử của A, ta ký hiệu $y \notin A$, đọc là: y không thuộc A.
- Có hai cách để biểu diễn một tập hợp:
  - Cách liệt kê: liệt kê tất cả các phần tử của tập hợp trong hai dấu ngoặc { }, hai phần tử cách nhau bởi một dấu phảy ",".

Ví dụ: X = {cam, quýt, ổi, xoài, me}; A = {1, 2, 3, 4}; ...

  - Cách đặc trưng: dùng một tính chất đặc trưng P để mô tả tập hợp: A={x{x thoả mãn P}, nghĩa là: mọi phần tử của A phải thoả mãn tính chất P và mọi phần tử thoả mãn tính chất P phải thuộc A.

Ví dụ: A = {x{ x là sinh viên khoá 41 trường ĐHBK}

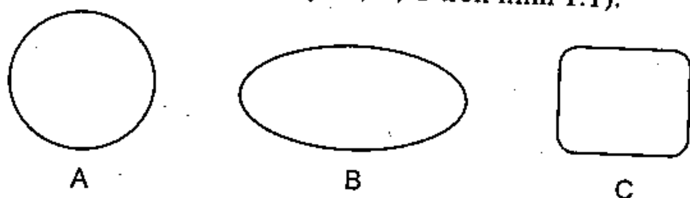
$B = \{m/n \mid m, n \in Z \text{ và } n \neq 0\}$.

- Tập hợp rỗng: một tập hợp không có phần tử nào được gọi là tập hợp rỗng, ký hiệu là ϕ.

  Ví dụ: $A = \{x \in R \mid x^2 + 2x + 3 < 0\} = \phi$

  $B = \{x \mid x$ là sinh viên khoá 38 trường ĐHBK và $x > 60$ tuổi$\} = \phi$

- Giản đồ Venn: Để biểu diễn trực quan trên giấy, người ta dùng phần mặt phẳng giới hạn bởi một đường cong kín để mô tả một tập hợp, bên cạnh ghi tên tập hợp ấy (Ví dụ: A, B, C trên hình 1.1).



Hình 1.1. Mô tả tập hợp

- Tập hợp con: Cho hai tập hợp A và B, ta nói B là tập con của A nếu và chỉ nếu mọi phần tử của B cũng là phần tử của A, ký hiệu: $B \subset A$

$$B \subset A \Leftrightarrow \forall x \in B \Rightarrow x \in A$$

- Hai tập hợp bằng nhau:

$$A = B \Leftrightarrow A \subset B \text{ và } B \subset A$$

1.1.2. Các phép toán về tập hợp

Cho hai tập hợp A và B, ta có các phép toán trên hai tập hợp như sau:

a) Phép giao: Giao của hai tập hợp A và B là một tập hợp, ký hiệu $A \cap B$ gồm những phần tử vừa thuộc A, vừa thuộc B.

$$A \cap B = \{x \mid x \in A \text{ và } x \in B\}$$

b) Phép hội: Hội của hai tập hợp A và B là một tập hợp, ký hiệu $A \cup B$, gồm những phần thuộc A hay thuộc B.

$$A \cup B = \{x \mid x \in A \text{ hay } x \in B\}$$

c) Phép hiệu: Hiệu của hai tập hợp A và B là một tập hợp, ký hiệu A\B, gồm những phần tử thuộc A và không thuộc B.

$$A \setminus B = \{x \mid x \in A \text{ và } x \notin B\}$$

Ví dụ: $A = \{1, 2, 3, 4, 5\}$; $B = \{2, 4, 6, 8\}$

$\Rightarrow A \cap B = \{2, 4\}$; $A \cup B = \{1, 2, 3, 4, 5, 6, 8\}$; $A \setminus B = \{1, 3, 5\}$

1.1.3. Tích Đề-các của các tập hợp

Cho hai tập hợp A và B, tích Đề-các của hai tập A và B là một tập hợp, ký hiệu $A \times B$, được định nghĩa:

$$A \times B = \{(x, y) \mid x \in A \text{ và } y \in B\}$$

Ví dụ: $A = \{1, 2, 3\}$; $B = \{a, b\}$

$\Rightarrow A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$

Cho n tập hợp $A_1, A_2, ..., A_n$, tích Đề-các của n tập hợp $A_1, A_2, ..., A_n$ là một tập hợp được ký hiệu và định nghĩa như sau:

$$\prod_{i=1}^{n} A_i = A_1 \times A_2 \times ... \times A_n = \left\{(x_1, x_2, ... x_n) \middle| x_i \in A_i, i = \overline{1..n}\right\}$$

Ví dụ: $A = \{1, 2, 3\}$; $B = \{a, b\}$; $C = \{\alpha, \beta\}$, lúc đó:

$A \times B \times C = \{(1, a, \alpha), (1, b, \alpha), (2, a, \alpha), (2, b, \alpha), (3, a, \alpha), (3, b, \alpha),$
$(1, a, \beta), (1, b, \beta), (2, a, \beta), (2, b, \beta), (3, a, \beta), (3, b, \beta)\}$

1.2. Quan hệ

1.2.1. Định nghĩa 1

Một quan hệ n ngôi trên một tập hợp $A \neq \phi$ là một tập con của A^n.

Ví dụ: Cho $A = \{1, 2, 3, 4\}$

$Q_1 = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$ là một quan hệ hai ngôi trên A.

$Q_1 = \{(1, 1, 1), (1, 2, 1), (2, 1, 2)\}$ là một quan hệ ba ngôi trên A.

1.2.2. Định nghĩa 2

Một quan hệ ngôi trên n tập hợp $A_1, A_2, ..., A_n \neq \phi$ là một tập con của tích Đề-các m tập hợp $A_{j1} \times A_{j2} \times ... \times A_{jm}$ với $A_{ji} \in \{A_1, A_2, ..., A_n\}$.

Ví dụ: Cho $A = \{1, 2, 3, 4\}$, $B = \{a, b, c\}$, $C = \{\alpha, \beta, \gamma, \delta\}$.

Lúc đó: $Q = \{(1, a, 2, \alpha), (2, b, 3, \beta), (4, a, 1, \gamma), (1, b, 4, \beta)\}$ là một quan hệ bốn ngôi trên A, B, C.

1.3. Ánh xạ

1.3.1. Định nghĩa 1 (ánh xạ)

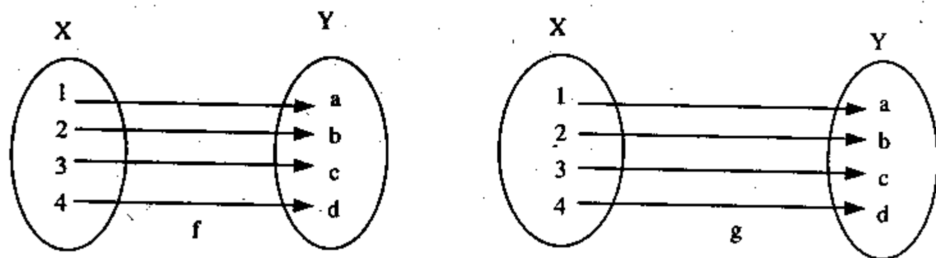
Cho tập hợp $X, Y \neq \phi$. Một ánh xạ f từ X vào Y là một quy luật tương ứng mỗi phần tử $x \in X$ với một phần tử $y \in Y$.

Ký hiệu $f: X \to Y$

$x \to y = f(x)$

$y = f(x)$ gọi là ảnh của x qua ánh xạ f.

Ví dụ 1: Cho X = {1, 2, 3, 4}, Y = {a, b, c, d} và các phép ứng f, g, h, k sau:



Hình 1.2. Ví dụ về ánh xạ của một tập hợp X đến tập hợp Y

Ta có: f, k là hai ánh xạ từ X vào Y, còn g, h không phải là hai ánh xạ từ X vào Y.

Ví dụ 2: Cho R là tập hợp các số thực và hai phép ứng f và g từ R vào R như sau: $f(x) = x^2 - 3x + 4$ và $g = 1/x \quad \forall x \in R$

Thì f là một ánh xạ, g không là một ánh xạ từ R vào R.

1.3.2. Định nghĩa 2 (ánh xạ hợp)

Cho ba tập hợp X, Y, Z ≠ ϕ và hai ánh xạ f: $X \to Y$ và g: $Y \to Z$ ta định nghĩa một ánh xạ từ X vào Z như sau:

$gof: X \to Z$

$x \longmapsto z = gof(x) = g[f(x)]$

gof(x) gọi là ánh xạ hợp của f và g, đọc là g tròn f.

Ví dụ: Cho f, g là hai ánh xạ: f(x) = sin(x), g(x) = e^x thì gof = $e^{Sin(x)}$, fog(x) = Sin(e^x)

1.3.3. Ánh xạ thu hẹp

Cho hai tập hợp X, Y ≠ ϕ, f là một ánh xạ X vào Y, $A \subset X$. Ta định nghĩa một ánh xạ: f_A: $A \to Y$ cho bởi $f_A(x) = f(x)\ \forall x \in A$, gọi là ánh xạ thu hẹp của f lên A.

Khi f_A là ánh xạ thu hẹp của X lên A thì ta cũng nói f là ánh xạ mở rộng của f_A lên X.

Ví dụ: Cho f: $R \to R$ như sau: $f(x) = x^2 + 1\ \forall x \in R$

và g: $N \to R$ như sau: $x(x) = x^2 + 1\ \forall x \in N$ thì $g = f_N$.

1.3.4. Định nghĩa 4 (ánh xạ chiếu)

Cho hai tập hợp A, B ≠ ϕ, X = A×B, khi đó ánh xạ Π_A: $X \to A$ được định nghĩa: $\Pi_A(a, b) = a\ \forall (a, b) \in X$, được gọi là ánh xạ chiếu của X lên A.

Cho $X = \prod_{i=1}^{n} A_{,i}$, $K = \prod_{j=1}^{m} A_{ij}$ với $A_{ij} \in \{A_1, A_2, \dots, A_n\}$, X, ta định nghĩa ánh xạ chiếu của X lên K như sau: Π_k: $X \to K$ với $\forall t = (a_1, a_2, \dots, a_n) \in X$, ta có:

$$\Pi_K(t) = t(K) = (a_{i1}, a_{i2}, \dots, a_{im}) \text{ với } a_{ij} \in A_{ij} \subset K.$$

Ví dụ: A = {1, 2, 3}, B = {a, b, c}, C = {x, y}; $X = A \times B \times C$, $K = B \times C$, lúc ấy :

$$\Pi_K(1, a, x) = (a, x); \Pi_K(1, a, y) = (a, y), \dots$$

II. CƠ SỞ DỮ LIỆU, HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

2.1. Một số ví dụ về cơ sở dữ liệu

Cơ sở dữ liệu là một khái niệm đầu tiên và chủ yếu liên quan tới bất kỳ hệ thống thông tin nào, nhất là đối với một hệ thống thông tin quản lý. Để bắt đầu, trước hết ta xét một vài ví dụ.

Ví dụ 1: Hệ thống bán vé máy bay của một hãng hàng không.

Để lựa chọn chuyến bay, khách hàng có thể tham khảo lịch bay của hãng, thông tin về các chuyến bay được lập theo bảng:

Mã chuyến bay	Loại máy bay	Sân bay đi	Sân bay đến	Ngày bay	Giờ bay	Giờ đến
VN272	ART72	SGN	NHA	30/06/02	7:20	08:05p
VN372	ART72	NHA	SGN	30/06/02	8:45	09:40p
VNA32	A320A	SGN	HAN	30/06/02	12:05	13:35p
VN472	ART72	SGN	DAK	30/06/02	12:25	13:25p
VNB77	BOE77	SGN	HAN	30/06/02	14:05	15:45p
VNB67	BOE67	HAN	SGN	30/06/02	14:05	15:45p
VNT06	TU106	HAN	DAN	30/06/02	16:25	17:35p
...	....	....	....	....	...	...

Mỗi chuyến bay là một mối quan hệ giữa các thuộc tính: Mã chuyến bay; Loại máy bay; Sân bay đi; Sân bay đến; Ngày bay; Giờ bay; giờ đến.

Bảng trên được gọi là một bảng dữ liệu, tuỳ từng đối tượng mà khai thác dữ liệu trong bảng trên theo mục đích của mình, chẳng hạn:

- Một khách hàng muốn bay từ Hà Nội vào Nha Trang, lúc đó anh ta chỉ quan tâm tới các dòng chứa thông tin về các chuyến bay từ Hà Nội tới Nha Trang.
- Trong khi đó đối với một nhà quản lý, ví dụ như Tổng Giám đốc hãng Hàng không Việt Nam, có thể ông ta chỉ cần biết tới số lượng chuyến bay thực hiện trong ngày bằng cách đếm số dòng trên bảng.
- ...

Điều này có nghĩa là các dữ liệu của bảng trên độc lập với các xử lý (về sau này là các chương trình khai thác nó) tác động lên chúng.

Dữ liệu trong bảng trên được tổ chức thành một bảng gồm các cột và các hàng. Các cột được gọi là lược đồ (SCHEME) hay bộ khung của bảng dữ liệu, mỗi cột được gọi là một thuộc tính (attribute) hay một trường, mỗi hàng (dòng) được gọi là một thể hiện (instance), một bộ (tuple) hay một bản ghi (record) của bảng dữ liệu.

Các thao tác thường áp dụng lên bảng dữ liệu là:

- Thêm một bản ghi.
- Xoá một bản ghi.
- Sửa một bản ghi.
- ...

Chú ý: Thao tác sửa một bản ghi thực chất là thực hiện hai thao tác: xóa bản ghi cũ rồi sửa vào bản ghi mới.

Ví dụ 2: Hệ thông tin thương mại

Trong một hệ thống thương mại, thông tin về các công ty thường được lưu trữ dưới dạng như bảng sau:

S#	Tên NCC	Trạng thái	Địa chỉ
S1	Anh Tú	25	HAN
S2	Thái Bình	10	SGN
S3	Tự Cường	30	SGN

Trong khi đó tình hình kinh doanh lại có thể mô tả nhờ vào bảng sau:

S#	P#	Số lượng
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Ý nghĩa của mỗi dòng trong bảng trên được hiểu như sau: Công ty Si bán được mặt hàng P*i* với số lượng là ...

Từ các ví dụ tương tự như vừa trình bày ở trên, ta có thể đưa ra một định nghĩa về cơ sở dữ liệu (CSDL) như sau:

2.2. Định nghĩa

Cơ sở dữ liệu là một tập những dữ liệu tác nghiệp của một đơn vị kinh doanh được lưu trữ lại và được các hệ ứng dụng của đơn vị sử dụng.

Ở đây thuật ngữ kinh doanh cần được hiểu theo nghĩa sau: kinh doanh là mang lại lợi ích, lợi nhuận hay giá trị thặng dư, như vậy mỗi trường học, mỗi bệnh viện, mỗi công ty, mỗi viện nghiên cứu,... đều là một đơn vị kinh doanh. Còn dữ liệu tác nghiệp là các dữ liệu liên quan tới hoạt động nghiệp vụ của đơn vị.

Một số tác giả khác lại đưa ra một định nghĩa tương đương như sau:

Cơ sở dữ liệu là một tập hợp các bảng dữ liệu có quan hệ với nhau sao cho cấu trúc của chúng cũng như các mối quan hệ bên trong giữa chúng là tách biệt với chương trình ứng dụng bên ngoài, đồng thời nhiều người dùng khác nhau cũng như nhiều ứng dụng khác nhau có thể cùng khai thác và chia sẻ một cách chọn lọc lúc cần.

Từ định nghĩa trên, ta thấy, một CSDL thoả mãn hai tính chất: tính độc lập dữ liệu và tính chia sẻ dữ liệu.

III. TÍNH ĐỘC LẬP DỮ LIỆU, CHIA SẺ DỮ LIỆU

Hai loại độc lập dữ liệu trong CSDL:

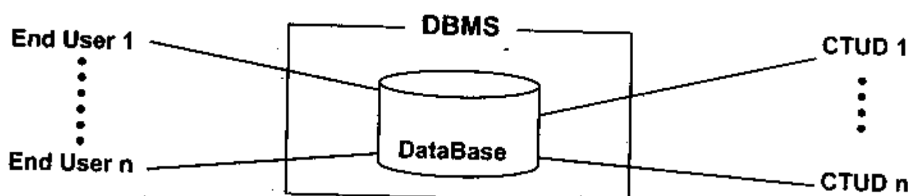
- Độc lập vật lý: thay đổi tổ chức của cơ sở dữ liệu vật lý có thể thay đổi hiệu quả tính toán của chương trình, nhưng không đòi hỏi phải thay đổi lại chương trình. Nói một cách khác, mức quan niệm phải đảm bảo được các truy nhập đến CSDL từ phía người sử dụng (NSD).
- Độc lập logic: sự thay đổi, thêm bớt thông tin về các thực thể ở mức quan niệm không đòi hỏi thay đổi các khung nhìn của NSD dẫn tới không cần thay đổi chương trình ứng dụng.

Tính chia sẻ dữ liệu: Vì độc lập với chương trình ứng dụng nên nhiều chương trình ứng dụng cùng sử dụng một cơ sở dữ liệu.

Với hai tính chất trên, ta có thể trừu tượng hoá dữ liệu ở mức cao. Sự trừu tượng hoá không những tăng cường hiệu quả quản lý mà còn giúp tư duy tốt về CSDL.

IV. HỆ CƠ SỞ DỮ LIỆU

4.1. Sơ đồ của một hệ CSDL (hình 1.3)



Hình 1.3: Sơ đồ một hệ CSDL

DBMS điều khiển hoạt động CSDL, nơi mà có thể có nhiều chương trình và nhiều người sử dụng có thể can thiệp vào.

4.2. Bốn thành phần của một hệ CSDL

1. CSDL hợp nhất

CSDL phải thoả mãn hai yêu cầu sau:

- Không dư thừa (trên thực tế là dư thừa ít nhất).
- Sử dụng chung.

2. Người sử dụng

Là người có yêu cầu truy nhập CSDL để thực hiện một thao tác nào đó:

- Người sử dụng cuối (End-User): là những NSD truy nhập vào CSDL từ một terminal, muốn tìm kiếm, tra cứu thông tin.
- Người viết chương trình ứng dụng: những người này ngoài những thao tác trên còn cần đến một ngôn ngữ lập trình (NNLT).
- Người quản trị CSDL: là người có nhiệm vụ điều khiển toàn bộ hệ CSDL, là người có quyền cao nhất, chịu trách nhiệm thực hiện các nhiệm vụ sau:

• Quyết định nội dung thông tin (dữ liệu nào sẽ được lưu trữ trong CSDL).

• Quyết định cấu trúc lưu trữ, chiến lược truy nhập.

• Xác định các phép thao tác đảm bảo tính đúng đắn và an toàn của dữ liệu.

• Xác định các phương án sao lưu.

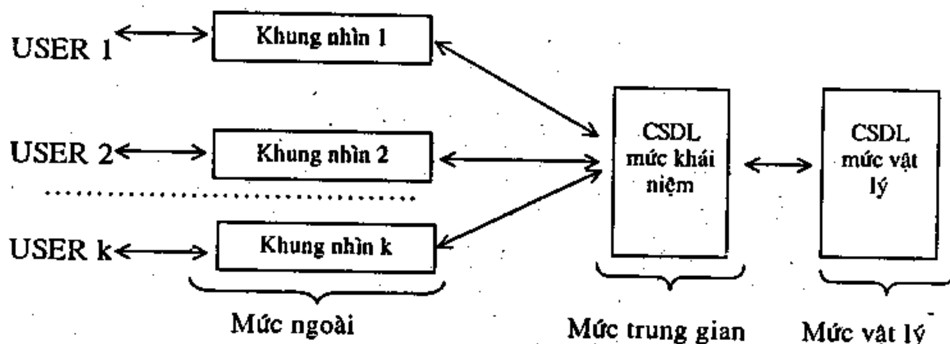
3. Hệ quản trị cơ sở dữ liệu

Đây chính là phần mềm của hệ CSDL.

4. Phần cứng

Là các thiết bị được sử dụng để lưu trữ dữ liệu.

4.3. Cấu trúc của một hệ cơ sở dữ liệu



Hình 1.4. Cấu trúc trúc của một hệ CSDL

Cấu trúc chuẩn gồm ba mức:

- Mức ngoài, là mức sát với người sử dụng nhất.
- Mức trung gian: không phụ thuộc vào user, không phụ thuộc vào sự lưu trữ dữ liệu.
- Mức lưu trữ vật lý: nơi lưu trữ dữ liệu trên các thiết bị nhớ.

Khung nhìn của một User là tập tất cả các dữ liệu mà User đó nhìn thấy phép truy nhập vào, là bộ phận của CSDL ứng với người đó.

4.4. Phân loại các hệ CSDL

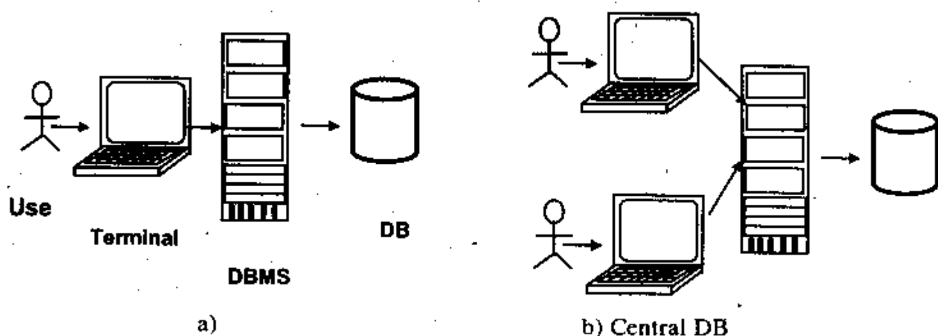
4.4.1. Các hệ tập trung

Là các hệ mà trong đó phần CSDL được lưu trữ tại một nơi, một vị trí nhất định.

a) Personal Database

Là một hệ CSDL nhỏ, bao gồm bốn thành phần (hình1.5a). Trong hệ này người viết chương trình chính là người sử dụng cuối, đồng thời lại là người quản lý luôn.

Một hệ như vậy đảm bảo một chức năng nhỏ, đơn lẻ (ví dụ: quản lý nhân sự ở một đơn vị hành chính).



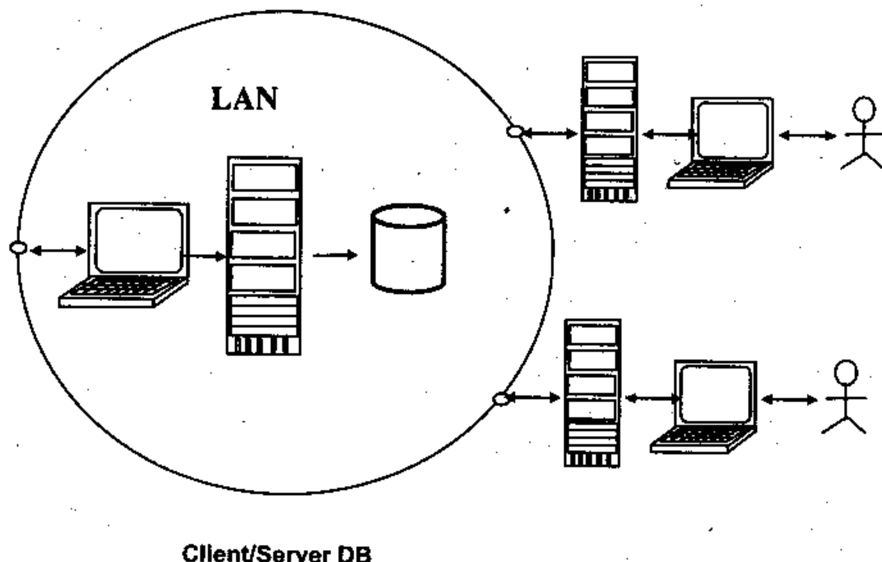
Hình 1.5. Hệ cơ sở dữ liệu
a) Personal DB; b) Central DB

b) Central Database

Là một hệ đa người dùng từ thiết bị đầu cuối (terminal) tại đó có màn hình và bàn phím để trao đổi thông tin. Mọi xử lý, tính toán được thực hiện tại trung tâm với một máy tính mạnh có thể xử lý nhiều yêu cầu.

Một hệ như vậy khi máy trung tâm có sự cố, thì toàn bộ hệ thống sẽ ngừng hoạt động.

c) Client/Server Database



Client/Server DB

Hình 1.6.

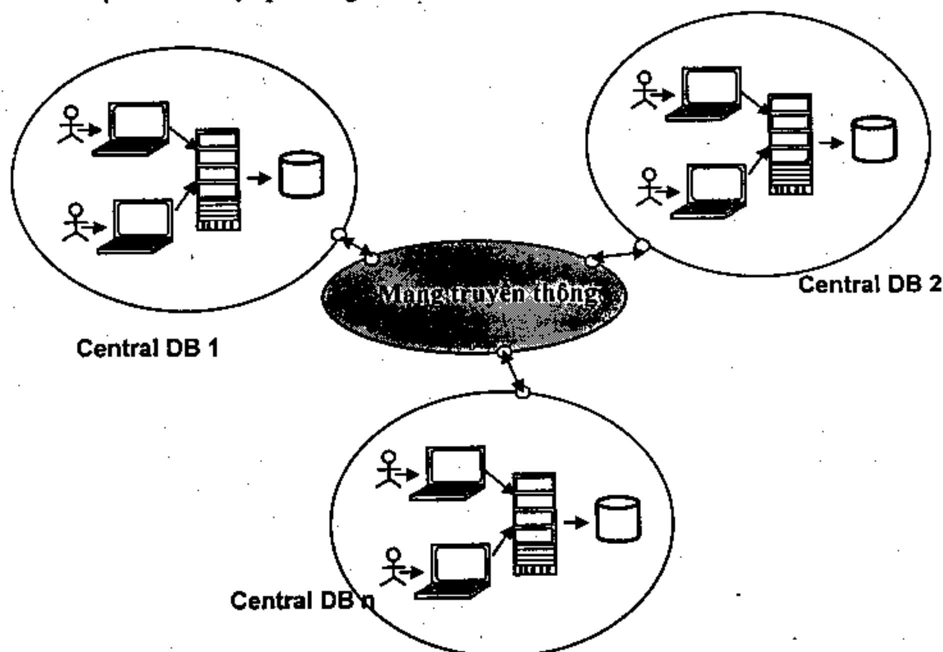
CSDL được tạo lập, lưu trữ tại Server, nhưng việc tính toán lại diễn ra ở máy Client. Máy Server không cần quá mạnh. Hệ CSDL loại này có vẻ "phân tán" nhưng thực ra vẫn là hệ tập trung vì dữ liệu đặt toàn bộ tại một chỗ (hình 1.6).

4.4.2. Các hệ CSDL phân tán

Là những hệ CSDL trong đó thành phần CSDL được chia nhỏ thành nhiều CSDL địa phương, trải ra trong một mạng máy tính.

Hình 1.7 mô tả một mạng truyền thông có ba trạm:

- Thành phần CSDL của hệ chia thành 3 CSDL địa phương.
- Hệ CSDL địa phương quản lý dữ liệu địa phương, phục vụ những người ở địa phương. Ví dụ trang quản lý ngân hàng: mỗi chi nhánh là một CSDL địa phương.



Hình 1.7. Hệ CSDL phân tán

Phân loại:

- Hệ thuần nhất.
- Hệ không thuần nhất.

Nếu các hệ CSDL địa phương biểu diễn theo những cách giống nhau, mô hình giống nhau, thì gọi là hệ thuần nhất, ngược lại gọi là hệ không thuần nhất.

4.5. Hai mức của hệ CSDL

- Mức logic hay mức khái niệm là mức giao tiếp giữa người sử dụng và mức vật lý.
- Mức vật lý: lưu trữ vật lý của cơ sở dữ liệu trên các thiết bị lưu trữ.
- Thể hiện (instance) : dữ liệu trong CSDL ở một thời điểm nhất định nào đó.

V. HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Một hệ quản trị cơ sở dữ liệu (DBMS) là một phần mềm điều khiển các truy nhập của người sử dụng đối với CSDL.

5.1. Các thao tác truy nhập chủ yếu

Các thao tác truy nhập chủ yếu gồm:

- Tìm kiếm dữ liệu theo một chỉ tiêu nào đó.
- Bổ sung dữ liệu vào CSDL (INSERT).
- Loại bỏ dữ liệu khỏi CSDL (DELETE).
- Sửa chữa dữ liệu trong CSDL (UPDATE).

5.2. Các bước hoạt động của một hệ quản trị CSDL

- Người sử dụng đưa ra các yêu cầu truy nhập dưới dạng các câu lệnh của một ngôn ngữ thao tác dữ liệu nào đó.
- DBMS nhận lời yêu cầu, phân tích cú pháp và chuyển cho mức logic.
- Mức logic tiến hành các truy nhập vào CSDL vật lý và trả lại kết quả.
- DBMS hiển thị kết quả cho người sử dụng.

5.3. Một số hệ DBMS sử dụng hiện nay

Quy mô lớn:

- ORACLE: chạy trên 80 cấu hình khác nhau.
- DB/2: một sản phẩm của IBM, đây là một trong những hệ quản trị cơ sở dữ liệu đầu tiên.
- INFORMIX.
- SYBASE.

Quy mô nhỏ:

- MS ACCESS.
- FOXPRO.

BÀI TẬP VÀ CÂU HỎI

1. Định nghĩa tích Đề-các. Cho các ví dụ minh hoạ.
2. Giới thiệu các phép toán trên tập hợp. Cho các ví dụ minh hoạ.
3. Định nghĩa cơ sở dữ liệu và các khái niệm liên quan.
4. Thế nào là hệ CSDL. Các thành phần của hệ CSDL.
5. Phân loại các hệ CSDL.
6. Trình bày khái niệm hệ quản trị CSDL? Hãy liệt kê các hệ quản trị CSDL mà anh (hay chị) biết.

Chương 2
CÁC MÔ HÌNH DỮ LIỆU

Ta có thể hiểu mô hình dữ liệu là một khuôn dạng của dữ liệu, cho phép người dùng nhìn thấy dữ liệu dưới cấu trúc thuật ngữ dễ hiểu mà ta gọi là lược đồ.

Một mô hình dữ liệu là một hình thức toán học bao gồm:

- Một hệ thống các ký hiệu để mô tả dữ liệu.
- Tập các toán tử dùng để thao tác trên dữ liệu này.

Vào những năm đầu của thập kỷ 60 (thế kỷ 20), mô hình mạng và mô hình phân cấp là thế hệ đầu tiên của gia đình các mô hình dữ liệu. Sang đầu thập kỷ 70, E.F. Codd đề xuất mô hình quan hệ mới, đó chính là thế hệ thứ hai. Mô hình quan hệ này có cấu trúc chặt chẽ, sáng sủa, nhất quán và có tính trực quan cao.

I. THỰC THỂ VÀ LIÊN KẾT

1.1. Thực thể và kiểu thực thể

Thực thể (entity) là đối tượng mà ta cần quan tâm trong công tác quản lý. Một đối tượng có thể là:

- Rất cụ thể như:
  - Nhân viên của một cơ quan là đối tượng cần quản lý.
  - Tờ hoá đơn là một đối tượng cần quản lý.
  - ...
- Hoặc trừu tượng, chẳng hạn:
  - Khoa tin học.
  - Ngành toán ứng dụng.
  - ...

a) Tiêu chuẩn xác định thực thể

- Có ích cho quản lý.
- Phân biệt được giữa các thực thể với nhau.

Ví dụ: Trong một nhà máy sản xuất đinh, rõ ràng là mỗi cái đinh có thể xem là một thực thể, tuy nhiên ở mức độ người quản lý thì nó không cần thiết để được xem là một thực thể vì người ta không thể quản lý tới từng cái đinh mặc dù nó rất cụ thể, trong trường hợp này đối tượng mà ta cần quản lý chính lại là các loại đinh:

- Đinh hai phân.
- Đinh ba phân
- ....

mặc dù chúng là trừu tượng.

b) Kiểu thực thể: Là tập hợp các đối tượng cùng loại hình thành một kiểu thực thể, nói khác đi kiểu thực thể chính là những thực thể cùng được mô tả bằng những đặc trưng giống nhau[1].

Ví dụ: một nhân viên là một thực thể, tập hợp các nhân viên của cùng một hệ thống tạo thành một kiểu thực thể.

c) Biểu diễn một kiểu thực thể: ta dùng một hình chữ nhật, bên trong ghi tên của kiểu thực thể:

Tên kiểu thực thể

Ví dụ:

Nhân viên

d) Thuộc tính: thuộc tính được hiểu là dữ liệu dùng để mô tả một đặc trưng của thực thể, ví dụ:

Tuổi của Tạ Tại Tuệ là 55.

Tổng tiền của một hoá đơn là 450.000 đồng.

...

Giá trị của một thuộc tính thường kèm theo một tên. Ví dụ tuổi: 45; giới tính: nam,...

Tên thuộc tính là tên chung cho một tập giá trị cùng kiểu, tên gọi chung đó được gọi là kiểu thuộc tính, ví dụ: tuổi, tổng tiền, giới tính,...

1.2. Liên kết

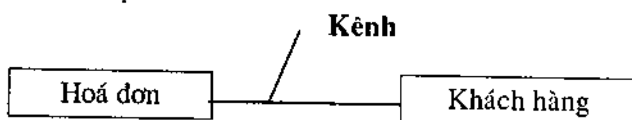
Một liên kết là một sự ghép nối giữa hai hay nhiều thực thể phản ánh một thực tế về quản lí.

Ví dụ:

Nhân viên Tạ Tấn thuộc Phòng Tổ chức;

Hoá đơn của Công ty Động lực,...

Cách biểu diễn liên kết: để biểu diễn sự liên kết của các thực thể người ta dùng một "kênh", ví dụ:

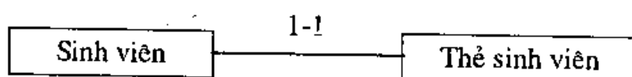


Kiểu liên kết: là tập hợp các liên kết cùng loại.

Phân loại liên kết:

- Liên kết *1-1* (đọc là liên kết một một): hai thực thể A và B có mối liên kết *1-1* nếu một thực thể kiểu A tương ứng với một thực thể kiểu B và ngược lại.

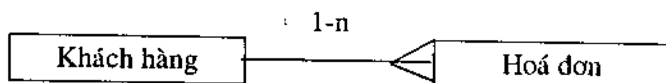
Ví dụ:



Ghi chú: Trong lược đồ cấu trúc dữ liệu, hai thực thể trong mối liên kết *1-1* sẽ được đồng nhất.

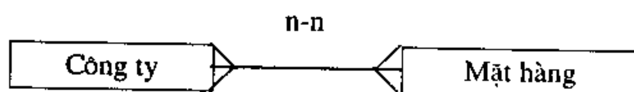
- Liên kết *1- n* (đọc là liên kết một nhiều): hai thực thể A và B có mối liên kết *1- n* nếu một thực thể kiểu A tương ứng với nhiều thực thể kiểu B và một thực thể của B chỉ tương ứng với một thực thể kiểu A.

Ví dụ:



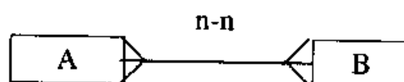
- Liên kết *n- n* (đọc là liên kết nhiều nhiều): hai thực thể A và B có mối liên kết *n-n* nếu một thực thể kiểu A tương ứng với nhiều thực thể kiểu B và ngược lại.

Ví dụ:

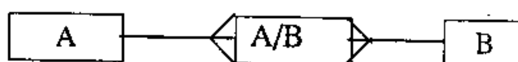


Ghi chú:

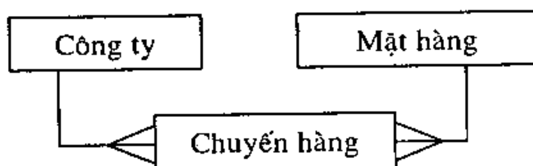
- Các liên kết *n – n* sẽ được thực thể hoá, chẳng hạn nếu ta có liên kết *n – n*:



Lúc ấy ta thể hiện lại mối liên kết nhiều nhiều này như sau:

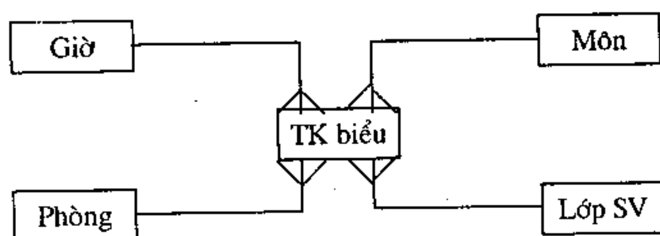


Ví dụ:



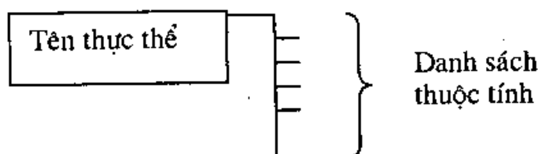
Nói một cách tổng quát, trong lược đồ cấu trúc dữ liệu người ta chỉ thể hiện mối liên kết *1-nhiều* giữa các thực thể.

- Liên kết còn thể hiện như sự kết nối giữa nhiều thực thể. Ví dụ: thời khoá biểu là một mối liên kết của nhiều thực thể với nhau:

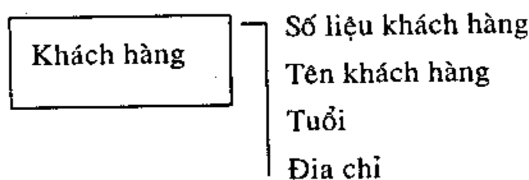


1.3. Biểu diễn đồ hoạ của một thực thể

Như đã biết, để mô tả một thực thể, người ta dùng một hình chữ nhật trong đó có ghi tên thực thể đó, tuy vậy cách mô tả này chỉ là ở bước đầu, một mô tả chính xác cần phải kèm theo các thuộc tính của thực thể nữa:



Ở đây *danh sách thuộc tính* được hiểu là tập hợp các thuộc tính, chẳng hạn:



Xét một phạm vi nhỏ trong công tác quản lý ở một nhà máy X, một cách cụ thể hơn là phạm vi theo dõi lao động của từng công nhân và phân xưởng

(một hệ thống con trong một hệ thống lớn). Để tiện cho việc biểu diễn, chúng ta sẽ mã hoá (viết tắt) các thuộc tính của các thực thể:

- Đối với công nhân :

  Số hiệu công nhân viết tắt SH_CN

  Họ tên CN ------- Tên CN

  Địa chỉ CN ------- DC_CN

  Bậc thợ ------- Bậc CN

  Chỉ số lương công nhân ------- Chỉ số

- Đối với phân xưởng :

  Số hiệu phân xưởng ------ SH_PX

  Tên phân xưởng -------- Tên_PX

  Trưởng phân xưởng -------- Trưởng PX

  Số lượng công nhân trong xưởng ------- Số lượng - CN

- Đối với máy :

  Số hiệu máy ------- SH_máy

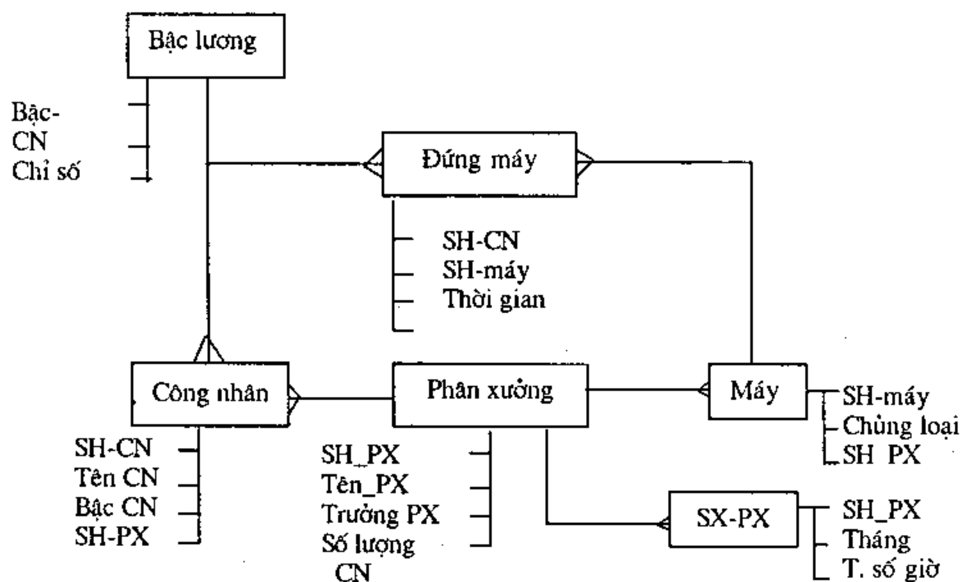
  Chủng loại máy ------- Chủng loại

- Đối với công lao động:

  Thời gian CN làm việc trong tháng trên một máy ------ Thời gian

  Tổng số giờ các máy đã chạy (trong 1 tháng) của máy --- Tổng số

Dựa vào các thực thể, các liên kết, các thuộc tính, ta thu được mô hình thực tế/liên kết (TT/LK) sau:



II. CÁC MÔ HÌNH DỮ LIỆU

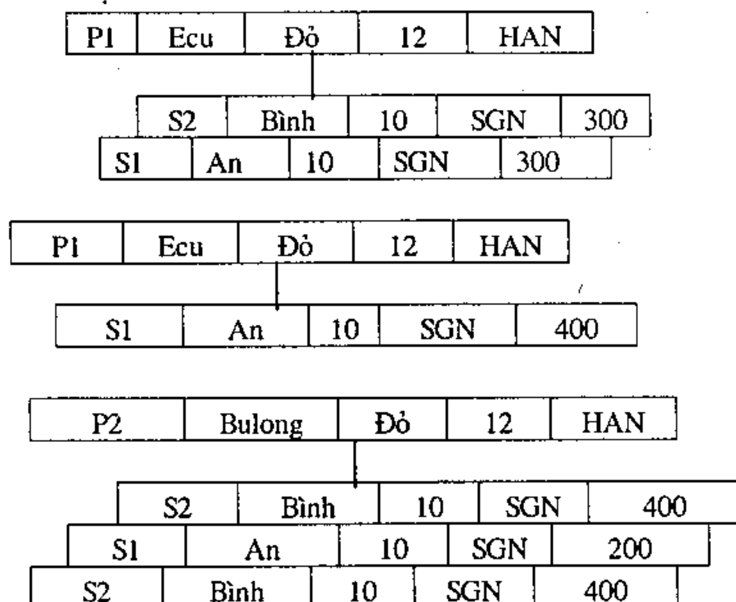
Cấu trúc của các thành phần trong một hệ CSDL chính là cốt lõi của nó. Như đã nói ở trên mô hình dữ liệu là một khuôn dạng của dữ liệu cho phép người dùng nhìn thấy dữ liệu dưới cấu trúc thuật ngữ dễ hiểu mà ta gọi là *lược đồ* (scheme).

Có ba hướng tiếp cận dữ liệu của CSDL là:

2.1. Mô hình phân cấp (Hierarchical model)

Mô hình phân cấp được đưa ra vào những năm 60, trong mô hình này dữ liệu được tổ chức thành cấu trúc cây, trong đó các nút (node) là tập các thực thể, các cành là mối quan hệ giữa hai nút theo mối quan hệ nhất định, cứng nhắc.

Ví dụ:



Trong trường hợp này, dữ liệu được biểu diễn dưới dạng cây đơn giản, trong đó các mặt hàng là gốc còn các nhà cung cấp là nhánh.

Các thao tác dữ liệu:

- Tìm kiếm: không đối xứng
- Bổ sung: Một hãng không cung cấp mặt hàng nào?
- Loại bỏ: loại bỏ một chuyến hàng dẫn đến loại bỏ cả hãng cung cấp.

- Cập nhật: nếu phải thay đổi thông tin của một hãng, phải dò cả mô hình sự xuất hiện của hãng đó.

2.2. Mô hình mạng (Network model)

Mô hình được đưa ra vào cuối những năm 60. Trong mô hình này dữ liệu được tổ chức thành một đồ thị có hướng, trong đó các đỉnh là các thực thể, các cung là quan hệ giữa hai đỉnh.

2.3. Mô hình quan hệ (Relational model)

Như đã biết, mô hình quan hệ được E.F Codd đưa ra vào đầu những năm 70, mô hình này dựa trên lý thuyết tập hợp và đại số quan hệ. Vì tính chất chặt chẽ của toán học về lí thuyết tập hợp nên mô hình này đã mô tả dữ liệu một cách rõ ràng, uyển chuyển và trở thành rất thông dụng.

Ngày nay hầu hết các hệ QTCSDL đều tổ chức dữ liệu theo mô hình dữ liệu quan hệ.

Trong mô hình quan hệ, dữ liệu được tổ chức thành các bảng (table), mỗi bảng tương ứng với một thực thể trong hệ thống.

Ví dụ: xét một *hệ thông tin phân phối hàng*, hệ này quản lý hoạt động cung ứng của một số nhà cung cấp (công ty) cung cấp các sản phẩm nhất định. Các thực thể chính của hệ thống bao gồm:

S (nhà cung cấp): mã nhà cung cấp (S#), tên, tình trạng, địa chỉ.

P (mặt hàng): mã mặt hàng (P#), tên, màu, khối lượng, nơi cất giữ.

SP (chuyến hàng): mã nhà cung cấp, mã mặt hàng, số lượng.

Ứng với mỗi thực thể ta có một bảng như sau:

Bảng S

S#	Tên NCC	Trạng thái	Địa chỉ
S1	An	25	HAN
S2	Bình	10	SGN
S3	Cường	30	SGN

Bảng P

P#	Tên	Màu	Khối lượng	Địa chỉ
P1	Ecu	Đỏ	12	HAN
P2	Bulông	Xanh	17	SGN
P3	Đai ốc	Xanh	17	DAN
P4	Đai ốc	Đỏ	14	HAN

Bảng SP

S#	P#	Số lượng
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	300
S2	P2	400
S3	P2	200

Về sau này:

- Một bảng như vậy gọi là một *quan hệ* (relation)
- Mỗi hàng gọi là *một bộ/ bản ghi.*
- Mỗi cột gọi là một *thuộc tính.*

Miền (Domain) là tập tất cả các giá trị có thể có của một cột nào đó.

Đặc điểm chủ yếu của cấu trúc dữ liệu là mối liên kết giữa các bộ (hàng) được biểu diễn duy nhất bằng các giá trị dữ liệu trong các cột rút ra từ một miền chung.

Xét một số thao tác dữ liệu cơ bản:

Tìm kiếm:

- Tìm những hãng cung cấp mặt hàng P2.
- Tìm những mặt hàng do hãng S2 cung cấp.

Bổ sung: bổ sung thêm một hãng cung cấp, một mặt hàng hay một chuyến hàng.

Cập nhật: thay đổi dữ liệu.

III. HỆ QUẢN TRỊ CSDL

3.1. Khái niệm

Hệ quản trị CSDL là một phần mềm thực hiện các công việc sau:

- Tạo lập và bảo tồn.
- Cho phép truy xuất đến CSDL theo thẩm quyền.
- Cập nhật dữ liệu.
- Bảo đảm an toàn và toàn vẹn dữ liệu.

3.2. Các chức năng của một hệ QTCSDL

Một hệ QTCSDL phải thực hiện được các chức năng sau:

- Tạo cấu trúc dữ liệu tương ứng với mô hình dữ liệu được chọn.
- Đảm bảo tính độc lập dữ liệu (dù có sửa dữ liệu thì chương trình cũng không cần sửa đổi theo).
- Tạo mối liên kết giữa hai kiểu mẫu tin có thể.
- Nạp dữ liệu vào CSDL.
- Cập nhật dữ liệu.
- Phát sinh các báo cáo từ các dữ liệu trong CSDL.
- Bảo tồn tính toàn vẹn dữ liệu trong CSDL.
- Bảo tồn tính an toàn dữ liệu trong CSDL.
- Cung cấp các tiện ích sao lưu, phục hồi (backup, recovery).
- Có các thủ tục điều khiển tương tranh (concurrency control).

3.3. Các thành phần của một hệ QTCSDL

Một hệ QTCSDL thông thường có các thành phần chính như sau:

- Ngôn ngữ định nghĩa dữ liệu (Data Definition Language).
- Ngôn ngữ thao tác dữ liệu (Data Manipulation Language).
- Ngôn ngữ hỏi (Query Language).
- Bộ viết báo cáo (Report Write).
- Bộ phát sinh đồ hoạ (Graphics Generator).
- Giao tiếp ngôn ngữ chủ (Host Language Interface).
- Ngôn ngữ thủ tục (Procedure Language).
- Từ điển dữ liệu.
- Bộ phát sinh ứng dụng.

BÀI TẬP VÀ CÂU HỎI

1. Thế nào là mô hình dữ liệu ?
2. Định nghĩa thực thể. Trình bày các khái niệm liên quan tới thực thể.
3. Hãy liệt kê các mô hình dữ liệu thông dụng mà anh (hay chị) biết.
4. Các chức năng và các thành phần của một hệ quản trị CSDL.

Chương 3

MÔ HÌNH QUAN HỆ

I. MỞ ĐẦU

Mô hình cơ sở dữ liệu quan hệ là một mô hình được sử dụng rộng rãi trong đời sống xã hội của mọi tổ chức, cơ quan, xí nghiệp, doanh nghiệp,... nói chung là ở bất cứ cơ quan nào cần quản lý và xử lý các thông tin. Hãy xét một vài ví dụ minh hoạ.

Ví dụ 1: Để thực hiện tốt công tác quản lý, bất cứ cơ quan nào đều lưu trữ hồ sơ cán bộ dưới dạng như sau:

MS	Họ tên	NS	Tr. độ	Quê quán	Giới tính	Lương
cb1	An	1942	Tiến sĩ	Thái Bình	Nam	5.24
cb2	Bình	1952	Đại học	Nam Định	Nam	4.47
cb3	Cam	1975	Đại học	Hà Nội	Nữ	3.89
cb4	Linh	1968	Tr. cấp	Hà Nam	Nữ	3.39
cb5	Tính	1958	Tr. cấp	Nam Định	Nam	5.21

Ví dụ 2: Sổ theo dõi khách của một khách sạn có thể có dạng sau đây:

MK	Đến	Đi	Phòng	Tiền
k1	02/01/04	04/01/04	201	300
k2	02/01/04	06/01/04	102	400
k3	03/01/04	05/01/04	302	250
k4	04/01	05/01/04	401	150

Để ý rằng, trong hai ví dụ trên, tuy quản lý các thông tin có bản chất khác nhau, nhưng cả hai đều có chung một đặc thù là: dữ liệu được trình bày dưới dạng bảng, mỗi bảng đều có dòng đầu chứa các tiêu đề cho các cột, các tiêu đề này về sau được gọi là các thuộc tính.

Chẳng hạn trong ví dụ 1, các thuộc tính là:

MS; Họ tên; NS; Tr. độ; Quê quán; Giới tính; Lương.

Còn trong ví dụ 2:

MK; Đến; Đi; Phòng; Tiền.

Mỗi thuộc tính đều có miền giá trị của nó, chẳng hạn thuộc tính *NS* có miền giá trị là các số nguyên: 1945, 1952, 1968,...Trong khi đó thuộc tính *Họ tên* lại có miền giá trị là các xâu ký tự: Nam, An, Bình,...

Trong mỗi bảng như vậy đều có một số phần tử, mỗi phần tử là một dòng. Về sau này các bảng như vậy gọi là một quan hệ.

II. CÁC KHÁI NIỆM CHÍNH

Trong mấy chục năm qua, các hệ cơ sở dữ liệu quan hệ đã thu được những thành tựu hết sức to lớn cả về phương diện lý thuyết và thực hành. Mô hình dữ liệu quan hệ đã trở thành một trong những mô hình dữ liệu có cơ sở lý thuyết được xây dựng vững chắc nhất. Do đó, việc nắm vững lý thuyết cơ sở của mô hình quan hệ nhằm ứng dụng vào việc thiết kế các hệ cơ sở dữ liệu quan hệ trong thực tế là rất cần thiết.

2.1. Miền, thuộc tính

Thuộc tính là dữ liệu mô tả một đặc trưng của một thực thể. Miền là một tập hợp các giá trị

Trong khuôn khổ giáo trình này, khái niệm miền luôn đi cùng với một thuộc tính nào đó.

2.2. Quan hệ

Quan hệ được hiểu như là là một tập con của tích Đề-các của một hay nhiều miền, như vậy về nguyên tắc mỗi quan hệ có thể là vô hạn, nhưng ở đây ta luôn giả thiết quan hệ là hữu hạn.

- Mỗi hàng (dòng) của quan hệ gọi là một bộ (typle).
- Quan hệ là một tập con của tích Đề-các D1 × D2 ×... × Dn gọi là quan hệ n - ngôi. Mỗi bộ của quan hệ có n thành phần và (n cột).
- Các cột của quan hệ gọi là các thuộc tính:

Định nghĩa:

Gọi U = {A1, A2,..., An} là tập hữu hạn các thuộc tính, mỗi thuộc tính Ai (i = 1,..., n) có miền giá trị tương ứng là dom (Ai). Người ta gọi r là quan hệ trên tập thuộc tính U nếu r là tập con của tích Đề-các của n miền dom (Ai):

$$r \subseteq dom(A1) \times dom(A2) \times \ldots \times dom(An).$$

Chú ý :

- Đương nhiên quan hệ r có thể bị thay đổi theo thời gian do việc thực hiện các phép toán cập nhật trên các bộ của quan hệ (bổ sung, loại bỏ,

sửa đổi,...), nói một cách rõ hơn, điều này có nghĩa là một quan hệ r còn là một hàm của thời gian.

- Để chỉ một quan hệ r trên tập thuộc tính {A1, A2,..., An}, ta dùng ký hiệu r(U) hay r(A1, A2,..., An).
- Để mô tả quan hệ r gồm p bộ có n thuộc tính, người ta dùng một bảng gồm n cột và p + 1 hàng, hàng thứ nhất là tên các thuộc tính, các hàng còn lại, mỗi hàng ứng với một bộ của quan hệ:

A_1	A_2	...	A_n
a_{11}	a_{21}	...	a_{n1}
a_{12}	a_{22}	...	a_{n2}
...	...	...	...

Ví dụ: Xét lược đồ quan hệ Lichbay30-06-02 (mã chuyến bay, loại máy bay, sân bay đi, sân bay đến, ngày bay, giờ bay, giờ đến).

Lúc đó một quan hệ Lichbay30-06-02 (lịch bay) có thể như sau:

Mã chuyến bay	Loại máy bay	Sân bay đi	Sân bay đến	Ngày bay	Giờ bay	Giờ đến
VN272	ART72	SGN	NHA	30/06/02	7:20	08:05
VN372	ART72	NHA	SGN	30/06/02	8:45	09:40
VNA32	A320A	SGN	HAN	30/06/02	12:05	13:35
VN472	ART72	SGN	DAK	30/06/02	12:25	13:25
VNB77	BOE77	SGN	HAN	30/06/02	14:05	15:45
VNB67	BOE67	HAN	SGN	30/06/02	14:05	15:45
VNT06	TU106	HAN	DAN	30/06/02	16:25	17:35

Biểu diễn một quan hệ:

Các thuộc tính của quan hệ Lichbay30-06-02 là: mã chuyến bay, loại máy bay, sân bay đi, sân bay đến, ngày bay , giờ bay, giờ đến.

Dom (MÃ CHUYẾN BAY) = tập các mã chuyến bay do Tổng Công ty Hàng không VN quy định.

Dom (LOẠI MÁY BAY) = tập các mã máy bay mà công ty hàng không đang có:

= {atr72, a320a, a320, boe77, boe67, boe47, tui06}.

Dom (SÂN BAY ĐI) = Dom(SÂN BAY ĐẾN) = tập các mã sân bay mà Công ty Hàng không có chuyến bay đi, đến.

Dom (NGÀY BAY) = Các ngày có chuyến bay

Dom (GIỜ BAY) = Giờ trong ngày.

Dom(GIỜ ĐẾN) = Giờ trong ngày.

Còn:

t = (VN272, ATR72, SGN, NHA, 30/06/02, 7:20, 08:05)

là một bộ của quan hệ Lichbay30-06-02.

Bộ này có ngữ nghĩa như sau: Chuyến bay số hiệu VN272 do loại máy bay ATR72 thực hiện từ Thành phố Hồ Chí Minh đi Nha Trang vào ngày 30 tháng 6 năm 2002, cất cánh lúc 7^h 20', tới Nha Trang vào lúc 8^h 05'.

Chú ý: Từ định nghĩa của quan hệ ta chú ý rằng:

- Thứ tự của các thuộc tính của lược đồ quan hệ không được xét đến.

Nghĩa là: $r = \{a, b, c\} \equiv r' = \{b, a, c\}$

- Trong một quan hệ, không có hai bộ giống nhau.

Trong chương này, để đơn giản trong việc biểu diễn, ta dùng các mẫu tự in hoa A, B, C,.... để chỉ tên các thuộc tính trong lược đồ quan hệ, còn a_i, b_j, c_k,... để chỉ các phần tử trong Dom(A), Dom(B), Dom(C),..., X, Y là một tập các thuộc tính của r (X, Y $\subset$ R).

2.3. Lược đồ quan hệ

Một lược đồ quan hệ (*relation scheme*) là một cặp có thứ tự:

S = < U, F>

Trong đó U là tập hữu hạn các thuộc tính của quan hệ và F là tập các ràng buộc của quan hệ. Ở đây một ràng buộc trên tập các thuộc tính $\{A_1, A_2,..., A_n\}$ được hiểu là một tính chất trên tập tất cả các quan hệ xác định trên tập thuộc tính này.

Chú ý:

- Để đơn giản khi nói tới một lược đồ quan hệ, khi mà không quan tâm tới các ràng buộc, ta ký hiệu một lược đồ quan hệ một cách đơn giản r(U) hay $r(A_1, A_2,.., A_n)$, với $\{A_1, A_2,.., A_n\}$ là tập thuộc tính.
- Lược đồ quan hệ mô tả cấu trúc của quan hệ. Tại mỗi thời điểm, lược đồ quan hệ có một thể hiện quan hệ cụ thể.
- Trên mỗi lược đồ quan hệ, có thể có nhiều quan hệ.

2.4. Thể hiện

Một thể hiện (relation instance) của lược đồ quan hệ S = < U, F> là tập tất cả các bộ thoả mãn tất cả các ràng buộc thuộc F.

Để tiện khi trình bày các khái niệm ta có quy ước sau: với một bộ t thuộc thể hiện r của lược đồ quan hệ S và $X \subseteq U$, $A \in U$ lúc đó:

- t[A] là giá trị của tại thuộc tính A.

- t[X] là bộ chỉ chứa các giá trị của các thuộc tính trong X.

2.5. Khoá của quan hệ và lược đồ quan hệ

2.5.1. Khoá

Định nghĩa 1

Cho r là một quan hệ định nghĩa trên lược đồ quan hệ <U,F>, với $U = \{A_1, A_2,..., A_n\}$, $K \subseteq U$, K được gọi là *khoá của quan hệ* r nếu:

$\forall t, t' \in r$ sao cho $t[K] = t'[K] \Rightarrow t \equiv t'$. (t[K] là phép chiếu của t lên K).

Nói một cách đơn giản là: K là khoá của r nếu và chỉ nếu không có hai bộ nào của r mà giá trị của chúng trên K là giống nhau.

Xét thêm rằng nếu $t1 \neq t2$ thì $t1\ [K] \neq t2\ [K]$, tức là giá trị trên K của một bộ nào đó khác mọi bộ còn lại, hay nói cách khác bộ đó là xác định duy nhất.

Ví dụ 2.1. Xét quan hệ:

r	(A	B	C	D)
t_1 =	a_1	b_1	c_1	d_1
t_2 =	a_1	b_1	c_2	d_2
t_3 =	a_2	b_1	c_2	d_2
t_4 =	a_2	b_1	c_2	d_2

Xét thấy: $K_1 = U$, $K_2 = ABC$, $K_3 = AB$ ... là các khoá của r

Nhưng X = BC không là khoá của r vì t 2 [X] = t_4[X] nhưng $t_2 \neq t_4$

Ví dụ 2.2. Xét quan hệ:

SV (MaSV, Ten SV, Namsinh, Malop)

K1 = {MaSV, Ten SV} là khoá

K2 = {Namsinh, TenSV} không là khoá

K3 = {Namsinh, TenSV, Diachi} không là khoá

K4 = {Malop} không là khoá

K5 = {MaSV} là khoá

K6 = {Malop, MaSV} là khoá

Định nghĩa 2

Cho lược đồ quan hệ <U,F>, với U = {A_1, A_2,...,A_n}, K ⊆ U, K được gọi là *khoá của lược đồ quan hệ* đang xét nếu K là khoá của mọi quan hệ r định nghĩa trên U.

Từ định nghĩa của khóa, ta thấy một quan hệ có ít nhất một khoá và đó chính là U.

2.5.2. Khoá tối thiểu

Định nghĩa 3

Cho r là quan hệ định nghĩa trên lược đồ quan hệ <U,F> với U = {A_1, A_2,...,A_n}; K ⊆U, K được gọi là *khoá tối thiểu* của quan hệ r nếu và chỉ nếu:

1. K là khoá của r.

2. Bất kỳ tập con thực sự K' ⊂ K nào đều không là khoá của r. Nghĩa là, nếu bỏ đi bất kỳ thuộc tính nào của K, thì phần còn lại không còn là khoá của r nữa.

Chẳng hạn xét lại ví dụ 2.1 như đã thấy:

K_2 = ABC không phải là *khoá tối thiểu* của r vì K_3 = AB ⊂ K_2 cũng là khoá của r.

K_3 = AB là *khoá tối thiểu* của r vì $K_3 \backslash A$ và $K_3 \backslash B$ đều không là khoá của r.

Còn trong ví dụ 2.2: K5 = {MaSV} chính là khóa tối thiểu.

Định nghĩa 4

Cho lược đồ quan hệ S = <U,F> với U = {A_1, A_2,...,A_n}, K ⊂ U, K được gọi là *khoá tối thiểu của lược đồ quan hệ* S nếu K là *khoá tối thiểu* của mọi quan hệ r định nghĩa trên lược đồ quan hệ trên.

Từ định nghĩa của khoá tối thiểu, ta thấy mọi quan hệ có ít nhất một khoá tối thiểu.

Vậy thì làm thế nào để xác định khoá cho một quan hệ ? Có rất nhiều thuật toán để thực hiện công việc này; sau đây là một trong số các thuật toán đó:

Bài toán: Cho một quan hệ r định nghĩa trên lược đồ quan hệ S = <U,F>, giả sử K ⊆ U, vậy trong điều nào K là khoá của r.

Thuật toán: Tính $\sigma_K(r)$. Nếu số bộ của $\sigma_K(r)$ bằng số bộ của r, thì K là khoá của r(U)

Chẳng hạn, xét quan hệ:

r(A	B	C)
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_1
a_2	b_2	c_1

K = AB là khoá vì :

r(K) = r' (A	B)
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_2

Xét một ví dụ khác:

	CANBO (MCB	*Ho_Ten*	*Nam_sinh*	*Noi_Lam_viec)*
t1	cb1	Lê Lựu	1940	Hội nhà văn
t2	cb2	Trần Tuệ	1950	Viện văn học
t3	cb3	Hoàng An	1942	Bộ Giáo dục và Đào tạo

Dễ nhận thấy rằng trong quan hệ CANBO trên MCB là khoá của quan hệ, vì mỗi giá trị của MCB đều xác định duy nhất một cán bộ trong quan hệ đang xét.

Lẽ dĩ nhiên trong một lược đồ quan hệ có thể có rất nhiều khoá, việc tìm tất cả các khoá của một quan hệ r là bài toán khó, cho đến nay vẫn chưa giải quyết được một cách trọn vẹn. Trong khuôn khổ giáo trình này, chúng tôi chỉ quan tâm đến khoá cho trước của lược đồ quan hệ, nghĩa là khi xây dựng một lược đồ quan hệ, ta chỉ định luôn các khóa của nó và trong quá trình xét các quan hệ trên lược đồ quan hệ này ta cũng chỉ quan tâm đến các khoá chỉ định mà thôi.

Chú ý: Nếu K là khoá của r thì mọi K' thoả mãn điều kiện $K \subseteq K'$ đều là khoá của r.

Khoá chính: Trong số các khoá tối thiểu của một quan hệ chỉ có một khoá được chọn ra làm khoá chính; những khoá còn lại là các khoá dự bị hay khoá phụ.

Khoá ngoài: Khoá ngoài không phải là khoá của quan hệ nhưng lại là khoá của một quan hệ khác.

III. CÁC PHÉP TÍNH TRÊN CƠ SỞ DỮ LIỆU QUAN HỆ

Các phép tính tác động lên một cơ sở dữ liệu bao gồm: chèn, loại bỏ, thay đổi,... Trong mô hình CSDL quan hệ, các phép tính này được áp dụng cho từng bộ của các quan hệ được lưu trữ trên máy.

Để thuận tiện khi giới thiệu các phép tính này, ta xét một quan hệ NHÂN VIÊN (Ho_Ten, Nam_sinh, Nơi_Lam_viec, Luong):

NHÂN VIÊN (Ho_Ten		Nam_sinh	Noi_Lam_viec	Luong)
t1	Lê Lựu	1940	Hội nhà văn	5.46
t2	Trần Tuệ	1950	Viện văn học	4.45
t3	Hoàng An	1942	Bộ GD và ĐT	4.42
t4	Trần Văn	1936	Bộ Y tế	6.65

3.1. Chèn thêm một bộ

Phép chèn thêm một bộ vào quan hệ r{A1,...,An}.

INSERT(r; A1=d1, A2=d2,..., An=dn)

Trong đó Ai (i=1,..,n) là tên các thuộc tính và di $\in$ dom(Ai).

Ví dụ: thêm một bộ t5 = (Vũ Văn Tiền, 1960, Bộ Tài chính, 4.28) vào quan hệ NHÂN VIÊN ở trên:

INSERT(NHÂN VIÊN; Ho_Ten = Vũ Văn Tiền, Năm_sinh =1960, Nơi_Lam_viec = Bộ Tài chính, Luong = 4.28).

Chú ý:

- Phép chèn có thể biểu diễn hình thức toán học như sau: $r = r \cup t$.
- Nếu xem thứ tự của các trường là cố định, khi đó ta có thể biểu diễn phép chèn một cách ngắn gọn hơn:

  INSERT(r; d1, d2,...,dn).

- Mục đích của phép chèn là thêm một bộ phận vào một quan hệ nhất định; kết quả của phép tính này có thể gây ra một số sai sót vì những lý do sau:
  - Bộ mới thêm vào không phù hợp với lược đồ quan hệ cho trước.

- Một giá trị của một thuộc tính nào đó nằm ngoài miền giá trị của thuộc tính đó.
- Giá trị khoá của bộ mới có thể là giá trị đã có trong quan hệ đang lưu trữ.

Do vậy, tuỳ từng hệ cụ thể sẽ có những cách khắc phục riêng.

3.2. Loại bỏ một bộ

Phép loại bỏ (DEL) là phép xoá một một bộ ra khỏi một quan hệ cho trước. Giống như phép chèn, phép loại bỏ có dạng biểu diễn hình thức toán học: r = r – t.

DEL(r; A1=d1, A2=d2,..., An=dn) hoặc ngắn gọn hơn có thể viết:

DEL(r; d1, d2,..., dn).

Ví dụ khi cần loại bỏ một bộ, chẳng hạn t2, từ quan hệ NHÂN VIÊN, ta viết:

DEL(NHÂN VIÊN; Trần Tuệ, 1950, Viện Văn học, 4.45).

Chú ý :Tất nhiên không phải lúc nào phép loại bỏ cũng cần đầy đủ thông tin về cả bộ cần loại. Khi ta có giá trị của bộ đó tại các thuộc tính khoá K = {B1,.., Bn}, lúc đó phép loại bỏ có dạng ngắn gọn sau:

DEL(r; B1=e1, B2=e2,...,Bi=ei).

3.3. Thay đổi giá trị các thuộc tính của một bộ

Khi cần điều chỉnh một số giá trị nào đó tại một số thuộc tính, lúc đó ta sử dụng phép thay đổi.

Gọi tập {C1,...,Cp} ⊆ {A1,...,An} là tập các thuộc tính mà tại đó các giá trị của bộ cần thay đổi; CH(r; A1=d1, A2=d2,..., An=dn; C1=e1, C2=e2,..., Cp=ep).

Nếu K ={B1,...,Bm} là khóa của quan hệ, khi đó chỉ cần viết:

CH(r; B1=b1, B2=b2,...,Bm=bm; C1=e1, C2=e2,...,Cp=ep).

Ví dụ: khi cần thay đổi lương của của ông Hoàng An trong quan hệ NHÂN VIÊN ta viết lệnh như sau:

CH(NHÂN VIÊN;Ho_Ten = Hoàng An, Luong =4.92).

Chú ý :

- Phép thay đổi là phép tính rất thông dụng; chúng ta có thể coi nó là tổ hợp của phép loại bỏ một bộ và phép chèn một bộ mới. Vì vậy phép *thay đổi* cũng phạm những sai sót như là phép loại bỏ và phép chèn.
- Phép thay đổi có dạng toán học: $r = r \backslash t \cup t'$.

BÀI TẬP VÀ CÂU HỎI

1. Hãy trình bày một vài ví dụ dẫn đến khái niệm quan hệ.
2. Định nghĩa quan hệ. Cho các ví dụ minh hoạ.
3. Định nghĩa khoá, khoá tối thiểu của một quan hệ. Cho ví dụ minh họa.
4. Trình bày các phép toán trên cơ sở dữ liệu quan hệ. Cho ví dụ minh họa.

Chương 4

NGÔN NGỮ ĐỊNH NGHĨA VÀ THAO TÁC DỮ LIỆU

I. ĐẠI SỐ QUAN HỆ

Ngôn ngữ đại số quan hệ là cơ sở quan trọng của một ngôn ngữ bậc cao được sử dụng để thao tác trên các quan hệ. Ngôn ngữ này bao gồm hai nhóm phép toán:

- Các phép toán tập hợp (phép giao, phép trừ, phép hợp và tích Đề-các).
- Các phép toán đặc biệt trên quan hệ (phép chọn, phép chiếu, phép kết nối và phép chia). Trước khi giới thiệu các phép toán này chúng ta hãy làm quen với một khái niệm mới:

Định nghĩa

Quan hệ khả hợp: hai quan hệ gọi là khả hợp nếu chúng cùng bậc (có số thuộc tính bằng nhau) và thuộc tính thứ i của quan hệ này có miền trị bằng miền trị thuộc tính thứ i của quan hệ kia.

Giả thiết r là quan hệ xác định trên tập thuộc tính U = {A1, A2,..., An}, với r là tập hữu hạn các bộ. Dưới đây là định nghĩa của các phép toán trên các quan hệ.

1.1. Phép hợp

Hợp của hai quan hệ r và s khả hợp, ký hiệu $r \cup s$ là tập tất cả các bộ thuộc r hoặc s hoặc thuộc cả hai quan hệ r và s.

Biểu diễn phép hợp có dạng:

$$r \cup s = \{ t \mid t \in r \text{ hoặc } t \in s \text{ hoặc } t \in r \text{ và } t \in s \}$$

Ví dụ:

R (A B C)	s (A B C)	$r \cup s$ = (A B C)
a_1 b_1 c_1	a_1 b_1 c_1	a_1 b_1 c_1
a_2 b_1 c_2	a_2 b_2 c_2	a_2 b_1 c_2
a_2 b_2 c_1	a_2 b_2 c_2	a_2 b_2 c_1

1.2. Phép giao

Giao của hai quan hệ r và s khả hợp, ký hiệu $r \cap s$ là tập tất cả các bộ thuộc cả r và s.

Biểu diễn hình thức:

$$r \cap s = \{ t \mid t \in r \text{ và } t \in s \}$$

Ví dụ: với r và s là hai quan hệ ở ví dụ trên, giao của chúng là:

```
         ( A B C )
r ∩ s = -------------
          a1 b1 c1
```

1.3. Phép trừ

Hiệu của hai quan hệ r và s khả hợp, ký hiệu r - s là tập tất cả các bộ thuộc r nhưng không thuộc s.

$$r - s = \{t \mid t \in r \text{ và } t \notin s\}$$

Ví dụ: cũng với r và s của ví dụ trên ta có:

```
          ( A B C )
r - s =  -------------
           a2 b1 c2
           a2 b2 c1
```

1.4. Tích Đề-các (Descartes)

Gọi r là quan hệ xác định trên tập thuộc tính $\{A_1, A_2,..., A_n\}$ và s là quan hệ xác định trên tập thuộc tính $\{B_1, B_2,..., B_m\}$. Tích Đề-các của r và s là tập (n + m) bộ sao cho n thành phần đầu có dạng một bộ thuộc r và m thành phần sau có dạng của một bộ thuộc s.

$r \times s = \{t \mid t$ có dạng $(a_1, a_2,..., a_n, b_1, b_2,..., b_m)$

trong đó : $(a_1, a_2,..., a_n) \in r$ và $(b_1, b_2,..., b_m) \in s\}$.

1.5. Phép chiếu

Phép chiếu là phép toán một ngôi, tác động lên một quan hệ.

Phép chiếu một quan hệ r trên tập các thuộc tính X của r, ký hiệu $\Pi_X r$ là một tập các bộ, được xây dựng bằng cách loại bỏ đi từ các bộ t trong quan hệ r những thuộc tính không nằm trong X. Thực chất của phép chiếu là loại

bỏ đi một số thuộc tính và giữ lại những thuộc tính còn lại của quan hệ đó. Để thuận tiện cho việc biểu diễn hình thức phép chiếu, quy ước một số ký hiệu như sau:

Gọi t là một bộ thuộc r, còn $A \in U$, lúc đó t[A] là giá trị của bộ t tại thuộc tính A. Giả sử $X \subseteq U$, với $X = \{B_1, B_2,..., B_m\}$ lúc đó $t[X] = (t[B_1], t[B_2],..., t[B_m])$.

Vậy $\Pi_X r = \{t[X] \mid t \in r\}$

Ví dụ: Cho X = AB và quan hệ:

r (A	B	C	D)
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_1
a_2	b_1	c_2	d_2
a_2	b_1	c_2	d_1

thì

$\Pi_X(r) =$ (A	B)
a_1	b_1
a_1	b_2
a_2	b_1

1.6. Phép chọn

Phép chọn là phép tính để xây dựng một tập con các bộ của quan hệ đã cho thoả mãn biểu thức q xác định.

Có thể diễn đạt như sau: *cho r là một quan hệ trên lược đồ quan hệ; một phép chọn trên r thoả mãn điều kiện q là một tập hợp được định nghĩa và ký hiệu:*

$\sigma_q(r) = \{t \mid \in r \mid q(t) = \text{đúng}\}$.

Biểu thức q được diễn tả bằng một tổ hợp Boolean của các toán hạng, mỗi toán hạng là một phép so sánh đơn giản giữa 2 biến là hai thuộc tính hoặc giữa một biến là một thuộc tính và một hằng, cho giá trị đúng sai đối với mỗi bộ đã kiểm tra.

Các phép so sánh trong biểu thức q là <, =, >=, <=, > và ≠. Các phép logic là ∧ (và), ∨ (hoặc) và ¬ (không).

Ví dụ: Cho q: A = a1 and B = b2, p: C = c1 và quan hệ:

r (A	B	C	D)
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_2	d_1
a_2	b_2	c_1	d_1

thì $\sigma_q(r)$

(A	B	C	D)
a_1	b_2	c_2	d_1
a_1	b_2	c_2	d_2

và $\sigma_p(r)$

(A	B	C	D)
a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_2

Phép kết nối

Khái niệm xếp cạnh nhau: Giả sử cho bộ d = (d1, d2,.., dn) và bộ e =(e1, e2,..., em). Phép xếp cạnh nhau của d và e được định nghĩa như sau: (d, e) = (d1, d2,..., dn, e1, e2,..., em).

Phép kết nối hai quan hệ thực chất là phép ghép cặp các bộ thoả mãn một điều kiện nhất định nào đó của hai quan hệ.

Chúng ta hiểu điều kiện kết nối hay biểu thức kết nối là phép hội của các toán hạng, mỗi toán hạng là một phép so sánh đơn giản giữa một thuộc tính của quan hệ r và một thuộc tính của quan hệ s. Phép kết nối của quan hệ r với quan hệ s với biểu thức kết nối F được định nghĩa như sau:

$$r \underset{F}{\bowtie} s = \{t \mid t = (u,v) \wedge u \in r \wedge v \in s \wedge F(t) = \text{đúng}\}$$

Tất nhiên ở đây cần giả thiết rằng các phép so sánh của các cặp thuộc tính thuộc hai quan hệ là có nghĩa, hay mỗi giá trị của thuộc tính này có thể so sánh được với mỗi giá trị của thuộc tính kia.

Các phép so sánh toán học được sử dụng là : =, >, <, ≠, ≥, ≤.

1.7. Phép kết nối tự nhiên

Trường hợp kết nối bằng tại thuộc tính cùng tên của hai quan hệ r và s và một trong hai thuộc tính đó được loại bỏ qua phép chiếu, thì phép kết nối đó được gọi là phép kết nối tự nhiên. (Hay nói một cách khác, kết nối bằng sau đó loại bỏ các thuộc tính trùng nhau (chỉ giữ lại một)

Cho $r_1(U_1)$ và $r_2(U_2)$. Gọi $S = U_1 \cap U_2$ và $U = U_1 \cup U_2$. Phép kết nối tự nhiên trên hai quan hệ r_1 và r_2 là một quan hệ r trên U được ký hiệu và định nghĩa:

$r_1 * r_2 = \{t(U) \mid \exists t_1 \in r_1$ và $\exists t_2 \in r_2$ và $t[U_1] = t_1, t[U_2] = t_2\}$

Nếu $S = \phi$ thì $r_1 * r_2$ là tích Đề-các thông thường.

Ví dụ 1:

r_1 (A	B	C)
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_2
a_2	b_3	c_1
a_3	b_3	c_2

r_2 (B	C	D)
b_1	c_1	d_1
b_1	c_2	d_2
b_2	c_1	d_1
b_2	c_2	d_2
b_3	c_1	d_1

$\Rightarrow r_1 * r_2$ (A	B	C	D)
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_1
a_2	b_1	c_2	d_2
a_3	b_3	c_1	d_1

Ví dụ 2:

r_1 (A	B)
a_1	b_1
a_1	b_2
a_2	b_1
a_2	b_3
a_3	b_3

r_2 (B	C)
b_1	c_1
b_1	c_2
b_2	c_1
b_2	c_2

$\Rightarrow r_1 * r_2$ (A	B	C)
a_1	b_1	c_1
a_1	b_1	c_2
a_1	b_2	c_1
a_1	b_2	c_2
a_2	b_1	c_1
a_2	b_1	c_2

Ví dụ 3:

r_1 (A	B)
a_1	b_1
a_1	b_2

r_2 (C	D)
c_1	d_1
c_1	d_2
c_2	d_1

$\Rightarrow r_1 * r_2$ (A	B	C	D)
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_1	c_2	d_1
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_2
a_1	b_1	c_2	d_1

1.8. Phép kết nối θ (theta join)

Trong phép kết nối tự nhiên, ta nhận thấy rằng, ta chỉ kết nối được những thể hiện (bộ) có giá trị bằng nhau trên những thuộc tính chung. Tuy nhiên, trong đời sống thực, có những trường kết nối không phải trên cùng những thuộc tính hay tất cả các thuộc tính chung.

Cho hai quan hệ $r_1(U_1)$, $r_2(U_2)$ là hai quan hệ trên U_1 và U_2, $X \in U_1$, $Y \in U_2$, θ là một trong các phép so sánh: <, ≤, >, ≥, ≠ giữa các giá trị của Dom(X) và Dom(Y). Phép nối kết θ giữa r_1 và r_2 là một quan hệ trên $R = R_1R_2$ được ký hiệu và định nghĩa như sau:

$$r = r_1 \frac{X}{X\theta Y} r_2(R) = \{t \in r(R) | t.X\theta t.Y\}$$

Ví dụ: θ là phép so sánh < và ≤:

r_1 (A	B)
1	b_1
2	b_2
3	b_1
1	b_3

r_2 (A	B)
1	d_1
1	d_2
2	d_1
2	d_2

$\Rightarrow r_1 \frac{X}{A \prec C} r_2$

(A	B	C	D)
1	b_1	2	d_1
1	b_1	2	d_2
1	b_3	2	d_1
1	b_3	2	d_2

$$\Rightarrow r_1 \underset{A \le C}{\overset{x}{—}} r_2$$

A	B	C	D
1	b_1	1	d_1
1	b_1	1	d_2
1	b_1	2	d_1
1	b_1	2	d_2
1	b_3	1	d_1
1	b_3	1	d_2
1	b_3	2	d_1
1	b_3	2	d_2
1	b_2	2	d_1
1	b_2	2	d_2

1.9. Phép chia

Gọi r là quan hệ n ngôi và s là quan hệ m ngôi ($n > m$, $s \neq \varnothing$). Phép chia r cho s, ký hiệu $r \div s$ là tập tất cả bộ_(n - m) ngôi, sao cho với mỗi bộ $u \in s$ thì $(t, u) \in r$.

$$r \div s = \{u \mid \forall v \in s: (u, v) \in r\}.$$

Ví dụ:

r(A B C D)	s(C D)	r ÷ s = (A B)
a b c d	c d	a b
a b e f	e f	e d
b c e f		
e d c d		
e d e f		
a b d e		

II. CÁC VÍ DỤ VỀ TÌM KIẾM BẰNG ĐẠI SỐ QUAN HỆ

Ví dụ ta có ba quan hệ:

S (#S, SNAME, STATUS, CITY) các công ty cung ứng.

P (#p, PNAME, COLOR, WEIGTH, CITY) các mặt hàng.

SP (#S, #P, QTY) các mặt hàng đã cung cấp.

- Tìm số hiệu những công ty đã cung cấp mặt hàng P2:

$$\Pi_{\#S} (\sigma_{\#P = 'P2'}(SP))$$

- Tìm số hiệu của những công ty cung ứng ít nhất là một mặt hàng màu đỏ:

$$\Pi_{\#S}(\sigma_{color = 'RED'}(P*SP)$$

- Tìm số hiệu những mặt hàng do công ty S2 cung ứng:

$$\Pi_{\#P}\,(\sigma_{\#S = 'S2'}(SP))$$

- Tìm số hiệu của những mặt hàng chưa được công ty nào cung ứng:

$$\Pi_{\#P}\,(P) - \Pi_{\#P}(SP)$$

III. NGÔN NGỮ HỎI ĐÁP DỮ LIỆU CÓ CẤU TRÚC (SQL)

Đây là ngôn ngữ định nghĩa và thao tác dữ liệu rất mạnh; ngôn ngữ này đã được chuẩn hoá và gọi là ANSI SQL. Tuy nhiên SQL của các hệ CSDL khác nhau cũng có những chi tiết khác nhau. Chúng ta sẽ trình bày các khả năng của ngôn ngữ SQL thông qua các câu hỏi cụ thể.

3.1. Ngôn ngữ định nghĩa dữ liệu

Trong ngôn ngữ SQL có một số phép tính để người sử dụng có thể tạo ra các quan hệ (bảng), các khung nhìn cũng như các tệp chỉ số. Đặc biệt, ở đây cũng cho phép xác định các thuộc tính được phép có giá trị không đổi, khi tạo một quan hệ bao gồm tên quan hệ, tên các thuộc tính, kiểu dữ liệu,...

3.1.1. Tạo bảng

Cú pháp như sau:

CREATE TABLE ten_bang(tên_cột kiểu [nonnull], ...)

ở đây: *ten_bang* là xâu ký tự không chứa ký tự trống, không trùng từ khóa;

tên_cột là xâu ký tự không chứa ký tự trống, trong một bảng không có hai cột trùng tên nhau.

kiểu: trong mệnh đề CREATE TABLE dùng một số loại dữ liệu sau:

integer số nguyên trong khoảng [-2 147 483 648 2 147 483 647],

smallinteger số nguyên trong khoảng [- 32 768 32 767],

decimal(n,p) ở đây n là số ký tự tối đa, p là số ký tự sau dấu '.',

float số dấu chấm động,

char(n) kiểu xâu ký tự có độ tự dài đúng n ký tự,

varchar(n) kiểu xâu ký tự có độ dài thay đổi và <= n ký tự,

date dữ liệu ngày tháng.

NULL là giá trị ngầm định, được dùng đến nếu cột không có giá trị cụ thể.

NOT NULL bắt buộc cột phải có giá trị cụ thể.

Ví dụ:

- Tạo quan hệ cơ quan (DEPT) gồm: mã số cơ quan với 2 ký tự không, chấp nhận giá trị không; tên cơ quan (gồm 12 ký tự); địa chỉ cơ quan (gồm 20 ký tự):

  CREATE TABLE DEPT (DNO CHAR(2) Not NULL,
  DNAME VARCHAR(12),
  LOC(VARCHAR(20))

3.1.2. Xóa bảng

Lệnh như sau DROP TABLE tên_bảng

3.1.3. Tạo tệp chỉ số

Trong SQL không có cơ chế tự động tạo tệp chỉ số cho các cột của bảng. Việc tạo chỉ số là do người sử dụng tự chọn. Mệnh đề tổng quát có dạng như sau:

CREATE [UNIQUE] INDEX tên_tệp_chỉ_số ON tên_bảng (tên_cột[ASC | DESC])

Chú ý ngầm định của hệ thống là ASC;

UNIQUE : trong số các giá trị bằng nhau chỉ lấy giá trị đầu.

Một số ví dụ:

- Tạo tệp chỉ số I3 trên thuộc tính CITY của quan hệ S:

  CREATE index I3 ON S(CITY)

- Tạo tệp chỉ số CS theo thứ tự tăng cho cột QTY của bảng SP

  CREATE index CS ON SP(QTY)

Trong SQL có thể tổ chức đa chỉ số, tức là tổ chức một tệp chỉ số cho nhiều cột, mỗi cột có thể có chiều tăng giảm khác nhau, thứ tự được tính từ trái qua phải; ví dụ: tạo tệp chỉ số I4 cho bảng SP theo cột #S tăng dần và cột #P giảm dần:

CREATE index I4 ON SP(#S ASC,#P DESC)

3.1.4. Tạo khung nhìn

Dạng tổng quát:

CREATE VIEW tên-view (danh sách tên cột) AS mệnh_đề_select.

Ví dụ tạo view PP gồm các cột #P, Pname từ bảng P của các mặt hàng màu đỏ:

```
CREATE view PP (#P,PNAME) as
Select #P,PNAME
From P
Where color = 'Đỏ'
```

Cũng như các phép tính tìm kiếm, một khung nhìn có thể thiết lập từ nhiều quan hệ khác nhau; tuỳ yêu cầu và quyền truy nhập dữ liệu của từng người sử dụng mà khung nhìn được thiết lập phù hợp, chẳng hạn : hãy tạo khung nhìn PPS bao gồm các cột #P, PNAME của các mặt hàng đã bán ra ít nhất một lần:

```
CREATE view PP (#P,PNAME) as
Select #P,PNAME
From P, SP
```

3.1.5. Thêm cột mới

Mệnh đề tổng quát như sau:

ALTER TABLE tên_bảng ADD tên_cột kiểu.

Ví dụ: thêm cột DONGIA cho bảng SP với kiểu số liệu là số thập phân:

ALTER TABLE SP ADD DONGIA decimal(8.2).

Một ví dụ khác : hãy bổ sung thuộc tính ngày cung cấp (NGAY) mặt hàng theo kiểu dữ liệu ngày tháng năm trong quan hệ SP.

ALTER TABLE SP ADD COLUMN NGAY date

3.1.6. Tạo liên kết

Tạo một tệp liên kết (link) L5 giữa các thuộc tính của quan hệ S và các thuộc tính của quan hệ SP thông qua thuộc tính #S

```
CREATE LINK L5
FROM S(#S)
TO SP(#S)
```

3.2. Ngôn ngữ thao tác dữ liệu

Ngôn ngữ thao tác dữ liệu bao gồm các khả năng sau:

3.2.1. Tìm kiếm dữ liệu

Câu lệnh tìm kiếm cơ bản là:

SELECT tc1, tc2,...

FROM tên_bảng1, tên_bảng2, ...

WHERE bth

Ở đây câu lệnh SELECT xác định các cột cần đưa ra kết quả.

FROM xác định các bảng cần lấy thông tin ra

WHERE xác định các bản ghi thỏa mãn yêu cầu chọn lọc để đưa ra kết quả.

Ngoài ra, để mở rộng khả năng của ngôn ngữ, khối SELECT còn được bổ sung thêm các mệnh đề *group by, having, order by*, các hàm mẫu và một số phần mềm còn có thêm cả mệnh đề *compute, for browse*, các phép toán tập hợp: *union, minus, intersec.*

1. Tìm kiếm đơn giản

a) Tìm kiếm không điều kiện

Ví dụ: Tìm các mặt hàng được bán ra:

SELECT #P

FROM SP.

b) Tìm kiếm với điều kiện đơn giản

Ví dụ tìm số hiệu và tên các công ty ở Hà nội:

SELECT #S, SNAME

FROM S

WHERE CITY = 'HAN'

Tất cả các bộ của quan hệ S sẽ được đem ra so sánh với giá trị 'HAN' tại thuộc tính CITY; những bộ thoả mãn điều kiện CITY = "HAN' được chiếu lấy các giá trị tại thuộc tính #S và SNAME.

Chú ý:

- *Nói chung, qua phép tính này, các bộ phận có thể trùng nhau mà không bị loại bỏ.*
- *Nếu muốn chỉ liệt kê một lần các bộ trùng nhau trên một cột nào đó ta dùng từ khoá DISTINCT.*

Ví dụ: Tìm các mặt hàng khác nhau được bán ra:

SELECT DISTINCT #P

FROM SP

- *Nếu muốn xem hết tất cả các cột, ta dùng dấu * thay cho các tên cột:*

Ví dụ: tìm tất cả các công ty ở Hà Nội

```
SELECT *
FROM S
WHERE CITY = 'HAN'
```

Trong biểu thức điều kiện WHERE có thể dùng các toán tử so sánh và logic như: =, >, <, >=, <=, AND, OR, NOT (tìm kiếm theo điều kiện).

Ví dụ: tìm số hiệu những công ty ở Hà Nội và có tình trạng lớn hơn 20:

```
SELECT #S
FROM S
WHERE CITY = 'HAN' AND STATUS >20
```

Ví dụ: tìm số hiệu các công ty đã bán mặt hàng P2:

```
SELECT DISTINCT #S
FROM SP
WHERE #P = 'P2'
```

Ví dụ: Tìm số hiệu các công ty đã cung ứng mặt hàng P2 với số lượng > 500.

```
SELECT DISTINCT #S
FROM SP
WHERE #P = 'P2' and QTY > 500
```

c) Tìm kiếm có xử lý xâu ký tự

Trong trường hợp người sử dụng không nhớ rõ tên người hoặc địa danh, lúc đó dùng từ khóa LIKE.

Ví dụ: tìm những mặt hàng chứa ở một tỉnh có tên bắt đầu bởi 'Hà':

```
SELECT *
FROM P
WHERE CITY LIKE 'Hà%'
```

Trong SQL dùng ký hiệu % thay cho một xâu con; còn '_' để thay cho một ký tự.

d) Tìm kiếm có xử lý ngày tháng

Ví dụ: giả sử trong SP có thêm cột ngày cung cấp hãy cho biết số hiệu các mặt hàng bán trước ngày 1/1/97:

```
SELECT #P
FROM SP
WHERE NGAY < {1/1/97}
```

e) Tìm kiếm với từ khóa IN và BETWEEN

Ví dụ: tìm số hiệu những công ty đã bán một trong ba mặt hàng P1, P2, P3:

```
SELECT DISTINCT #P
FROM SP
WHERE #P IN ('P1', 'P2', 'P3')
```

Ví dụ: tìm số hiệu những mặt hàng có khối lượng từ 100 đến 300:

```
SELECT #P
FROM P
WHERE weigth BETWEEN 100 and 300
```

2. Các hàm thư viện của SQL

Trong ngôn ngữ SQL có một danh sách các hàm thư viện là COUNT, SUM, AVG, MAX, MIN và SET.

COUNT : tính số lượng các bộ hiện có trong một quan hệ đang quan tâm.

MAX: chọn giá trị cực đại của một trường trong một quan hệ.

MIN: chọn giá trị cực tiểu của một trường trong một quan hệ.

AVG (average): tính giá trị trung bình của một trường trong một quan hệ.

SUM(tên_cột) : tính tổng giá trị xuất hiện trên cột.

SET : tập.

Trừ hàm SET chỉ dùng trong tân từ, các hàm khác vừa dùng trong tân từ vừa dùng trong mệnh đề SELECT, ví dụ:

- Tính tổng số các công ty :

```
SELECT COUNT(#S)
FROM S
```

- Tính tổng số những công ty đã bán mặt hàng P2:

```
SELECT COUNT (#S)
FROM SP
WHERE #P = 'P2'
```

Chú ý: Hàm COUNT khi có đối số là '*' có nghĩa là đếm số bản ghi thỏa mãn yêu cầu tìm kiếm mà không đề cập tới bất cứ cột nào.

Ví dụ: Cho biết số lần mặt hàng P2 đã được bán ra:

```
SELECT COUNT (*)
FROM SP
WHERE #P = 'P2'
```

- Tìm tổng lượng mặt hàng P2 đã bán ra:

```
SELECT SUM(QTY)
FROM SP
WHERE #P = 'P2'
```

- Tìm hiệu số mặt hàng P1 bán một lần nhiều nhất và một lần ít nhất của công ty S1:

```
SELECT MAX(QTY) – MIN(QTY)
FROM SP
WHERE #S ='S1' AND #P = 'P1'
```

- Cho biết số lượng mã số mặt hàng khác nhau đã được bán ra:

```
SELECT COUNT(DISTINCT #P)
FROM SP
```

- Đối với mỗi mặt hàng đã được bán ra, tìm số liệu của nó và tính tổng số công ty khác nhau đã đưa ra thị trường mặt hàng đó:

```
SELECT #P, COUNT ((DISTINCT #S)
FROM SP
GROUP BY # P
```

Ở đây toán tử GROUP BY phân hoạch quan hệ SP theo #P, sau đó SELECT tìm các giá trị # P đơn nhất và đếm lần lượt cho từng phần tất cả những giá trị #P tương ứng.

3. Tìm kiếm có sắp xếp

Đối với mỗi mặt hàng đã được bán ra, tìm số hiệu của nó và tính tổng số công ty khác nhau đã đưa ra thị trường mặt hàng đó:

```
SELECT #P, COUNT(DISTINCT #S)
FROM SP
GROUP BY #P
```

Chú ý:

- Sau ORDER BY là tên cột rồi đến chiều sắp xếp tăng hoặc giảm.
- Có thể sắp xếp nhiều cột và nếu không chỉ ra chiều sắp xếp thì ngầm định là ASC.
- Mệnh đề ORDER BY nếu đứng sau GROUP BY thì miền tác động của sắp xếp là trong từng nhóm của cột được chỉ ra trong GROUP BY.

- Nếu cột sắp xếp có mặt ở mệnh đề SELECT thì trong mệnh đề ORDER BY chỉ cần chỉ ra số thứ tự của cột đó trong danh sách tham chiếu là đủ; chẳng hạn ví dụ trên có thể viết lại như sau:

```
SELECT PNAME, #P
FROM P
WHERE COLOR = 'ĐỎ'
ORDER BY 2 DESC
```

4. Tìm kiếm với câu hỏi phức tạp

Trong phần này trình bày việc tìm kiếm trên nhiều bảng qua việc sử dụng ánh xạ lồng hoặc qua phép kết nối.

- Phép kết nối

Thực hiện phép kết nối thông qua các cột, các cột tham gia kết nối phải có miền trị sánh được với nhau; tên cột của các bảng khác nhau có thể viết tường minh thông qua tên bảng.

Ví dụ: với mỗi mặt hàng đã được bán ra, cho biết mã số mặt hàng và địa chỉ của công ty đã cung cấp mặt hàng đó:

```
SELECT #P,CITY
FROM SP, S
WHERE SP.#S = S.#S
```

Ví dụ: tìm tên các công ty khác nhau đã bán hàng có địa chỉ ở Hà Nội:

```
SELECT DISTINCT SNAME
FROM S,SP
WHERE S.#S = SP.#S AND CITY = 'HAN'
```

Ví dụ: cho biết tên của các công ty đã bán mặt hàng P2:

```
SELECT SNAME
FROM S,SP
WHERE SP.#S = S.#S AND #P = 'P2'
```

- Ánh xạ lồng (Các câu hỏi lồng nhau; tìm kiếm sử dụng ánh xạ lồng nhau).

Ví dụ: tìm tên những công ty đã đưa ra thị trường mặt hàng 'P2'.

```
SELECT SNAME
 FROM S
 WHERE #S IN (SELECT #S
                     FROM SP
                    WHERE #P = 'P2')
```

Ở đây IN là toán tử kiểm tra phép thuộc về (∈) trong tập hợp, ngoài ra SQL còn có các toán tử khác như NOT IN, CONTAINS, DOES NOT CONTAIN; các toán tử tập hợp như: UNION, INTERSECT, MINUS.

Phép lồng nhau cũng có thể được lồng theo nhiều mức: hoặc sử dụng ánh xạ lồng nhau theo mức hoặc sử dụng ánh xạ lồng nhau với đường dẫn giữa các khối khi mỗi khối hướng tới một quan hệ khác nhau (phép lồng cũng có thể tham chiếu từ câu hỏi con lên câu hỏi chính).

Ví dụ: tìm tên những công ty không bán mặt hàng P1:

```
SELECT SNAME
FROM S
WHERE #S NOT IN
        SELECT #P
        FROM SP
        WHERE #P= 'P1'
```

• Tìm kiếm có chứa phép tính tập hợp

Ví dụ: tìm số hiệu của những công ty hiện thời chưa bán được một mặt hàng nào cả:

```
(SELECT #S FROM S)
MINUS
(SELECT #S FROM SP)
```

• Tìm có chứa phép so sánh tập hợp

Ví dụ: tìm tên những công ty cung cấp tất cả các mặt hàng:

```
SELECT SNAME
FROM S
WHERE (SELECT #P FROM SP
        WHERE #S = S.#S)
                =
                (SELECT #P
                FROM P)
```

• Tìm kiếm chứa GROUP BY, HAVING, SET

Tìm mã số những mặt hàng mà mỗi công ty đã bán cho khách hàng:

```
SELECT #P
FROM SP
GROUP BY #S
```

Trong mệnh đề này bảng SP được lấy ra, sau đó phân thành nhóm theo mã số của công ty (#S); có nghĩa là các bộ có cùng giá trị #S sẽ sắp xếp liên tiếp nhau hết nhóm này đến nhóm khác.

Mệnh đề HAVING được sử dụng cùng với GROUP BY; sau HAVING là biểu thức điều kiện. Biểu thức này không tác động vào toàn bảng được chỉ ra ở mệnh đề from mà chỉ tác động lần lượt vào từng nhóm các bản ghi đã chỉ ra tại mệnh đề GROUP BY.

Ví dụ: tìm mã số những công ty đã bán ít nhất hơn hai mặt hàng

```
SELECT #S
FROM SP
GROUP BY #S
HAVING count (DISTINCT #P)>2
```

Ví dụ: tìm số hiệu những công ty nào bán ít nhất là tất cả những mặt hàng do công ty S2 cung ứng:

```
SELECT #S
FROM SP
GROUP BY #S
HAVING SET (#P) CONTAINS
(SELECT #P
FROM SP
WHERE #S = 'S2')
```

GROUP BY chia nhóm sao cho bên trong mỗi nhóm có các dòng đều cùng chứa một giá trị như trong thuộc tính chỉ ra (#S).

HAVING là mệnh đề đặc biệt cho tập SET.

Thao tác:

Từng nhóm SET(#P) so sánh với tập trong dấu ngoặc.

- Tìm kiếm có chứa một tập hằng

Ví dụ: tìm số hiệu của những hãng cung ứng cả hai mặt hàng P1 & P2:

```
SELECT #S
FROM SP
GROUP BY #S
HAVING SET(#P) CONTAINS ('P1', 'P2')
```

3.2.2. Các phép tính cập nhật

Trong SQL có ba phép cập nhật, nhưng không được thao tác đồng thời trên một quan hệ.

1. Phép sửa đổi (UPDATE)

Ví dụ: đổi màu sắc của mặt hàng P2 thành màu vàng:

```
UPDATE P
SET COLOR = 'Yellow'
WHERE #P = 'P2'.
```

Cú pháp tổng quát như sau:

```
UPDATE tên_bảng
SET tc1 = gtr1, tc2 = gtr2,....
WHERE biểu thức điều kiện
```

Xét một ví dụ khác, giả sử rằng quan hệ P có chứa một thuộc tính phụ là QOH; công ty S1 hiện đang cung cấp mặt hàng P1 nhiều hơn trước là 10 đơn vị. Hãy thêm 10 vào lượng hàng hiện có cho P1 và vào lượng P1 do S1 cung ứng; khi đó lệnh được viết như sau:

```
UPDATE P
SET QOH = QOH +10
WHERE #P = 'P1'
UPDATE SP
SET QTY = QTY +10
WHERE #S = 'S1' AND #P = 'P1'
```

2. Phép bổ sung (INSERT)

Cú pháp tổng quát như sau:

```
INSERT INTO tên bảng
VALUES (giá trị các cột)
```

Ví dụ: bổ sung mặt hàng P4 tên là 'vải' màu 'vàng', khối lượng 2, thành phố 'Hà Nội' vào bảng P:

```
INSERT INTO P
< 'P4', 'vải', 'vàng', 2, 'HAN'>.
```

Ghi chú:

- Câu lệnh trên chỉ bổ sung một bản duy nhất; nếu muốn bổ sung nhiều bản ghi, lần lượt viết tập các bản ghi cần bổ sung như trên.
- Tổng quát hơn ta có thể bổ sung một tập các bản ghi là kết quả của việc xử lý câu hỏi nào đó, chẳng hạn:

```
INSERT INTO P
SELECT *
```

```
FROM W
WHERE COLOR = 'đỏ'
```

Dĩ nhiên ta phải giả sử rằng hai quan hệ W và P là tương hợp.

3. Phép loại bỏ (DELETE)

Cú pháp:

```
DELETE tên bảng
WHERE biểu thức điều kiện.
```

Ví dụ: loại bỏ công ty S1 ra khỏi quan hệ S:

```
DELETE S
WHERE #S = 'S1'
```

3.2.3. An toàn dữ liệu

Ngôn ngữ SQL cho phép người sử dụng kiểm tra dữ liệu của mình khi cập nhật và tuyên bố quyền truy nhập tới cơ sở dữ liệu để đảm bảo cho tính nhất quán và toàn vẹn dữ liệu. Đặc biệt trong trường hợp có nhiều người cùng sử dụng hệ thống, nhất là cùng truy nhập tới cùng một tệp (bảng) của cơ sở dữ liệu.

Vì vậy cần phải có những biện pháp phòng ngừa để ngăn ngừa các nguy cơ:

- Vô tình sử dụng sai.
- Sự cố trong quá trình xử lý giao dịch.
- Dị thường gây ra bởi truy cập đồng thời vào CSDL.
- Dị thường gây ra bởi sự phân tán của dữ liệu trên một số máy tính.
- Cố tình sử dụng sai.
- Đọc dữ liệu một cách trái phép (hiểu là đánh cắp thông tin).
- Sửa đổi dữ liệu trái phép.
- Phá hoại dữ liệu trong CSDL.
- ...

Bảo vệ dữ liệu trên hai phương diện:

- *An toàn CSDL*: chỉ việc bảo vệ CSDL tránh khỏi những hiện tượng cố tình sử dụng sai dữ liệu.
- *Toàn vẹn CSDL*: chỉ việc tránh khỏi những sự vô tình hay cố ý làm mất tính nhất quán của dữ liệu.

Sau đây là một số biện pháp.

Biện pháp 1: Sự cấp quyền

Nói một cách cụ thể người sử dụng có thể có một số trong số những *quyền truy cập* sau:

- *Quyền đọc*: được phép đọc dữ liệu trong CSDL.
- *Quyền chèn thêm dữ liệu*: được phép chèn dữ liệu mới vào trong CSDL có sẵn nhưng không được thay đổi bất kỳ dữ liệu có sẵn nào.
- *Quyền cập nhật*: được phép sửa đổi dữ liệu nhưng không được xoá dữ liệu.
- *Quyền xoá*: được phép xoá dữ liệu trong CSDL.

Ngoài quyền truy nhập, người sử dụng còn có thể được phép sửa đổi lược đồ CSDL.

- *Quyền tạo chỉ mục*: được phép tạo các chỉ mục.
- *Quyền quản lý tài nguyên*: được phép tạo các quan hệ mới.
- *Quyền thay đổi*: được phép thêm hoặc xoá các thuộc tính trong quan hệ.
- *Quyền loại bỏ*: loại bỏ một quan hệ.

Biện pháp 2: Quyền tạo và sử dụng khung nhìn (views)

Khung nhìn là một phương thức cho phép:

- Che dấu những dữ liệu mà người sử dụng cụ thể nào đó không cần thiết phải "nhìn" thấy.
- Làm đơn giản hoá việc sử dụng hệ thống.
- Làm tăng cường an toàn hệ thống.
- ...

Quyền tạo và sử dụng khung nhìn cũng phụ thuộc vào quyền truy nhập hay sửa đổi của người dùng.

Biện pháp 3: Sự cấp đặc quyền luân chuyển dữ liệu.

Một người sử dụng có thể chuyển quyền cho những người sử dụng khác và cần phải kiểm tra cẩn thận những đặc quyền này.

Sơ đồ chuyển quyền

Những câu lệnh cấp và thu hồi quyền trong SQL:

- Lệnh cấp quyền: Việc tuyên bố và kiểm tra quyền truy nhập CSDL được thực hiện qua mệnh đề GRANT. Cú pháp như sau:

GRANT *danh_sách_quyền*
ON *đối_tượng*
TO *danh_sách_người_sử_dụng* [WITH GRANT OPTION];

Các quyền truy nhập trong SQL bao gồm: read, select, write, insert, update, delete và run.

đối_tượng: là tên bảng/tên khung nhìn/tên chương trình.

danh_sách_ người_ sử_ dụng: là tên người hoặc một nhóm người.

Từ khoá WITH GRANT OPTION đảm bảo cho người sử dụng có thể tiếp tục trao quyền cho người khác khi cần.

Ví dụ 1:

GRANT read ON S TO An WITH GRANT OPTION.

Câu lệnh này trao quyền đọc bảng S cho An và khi cần thì An có thể trao quyền đó cho người khác.

- Câu lệnh thu hồi quyền: REVOKE, cú pháp như sau:

REVOKE *danh_sách_quyền*

ON *tên bảng/tên khung nhìn/tên chương trình*

FROM *danh_sách_người_sử_dụng*

Việc huỷ bỏ quyền của một người sử dụng kéo theo việc huỷ bỏ quyền của những người sử dụng được uỷ quyền.

Biện pháp 4: Mã hoá dữ liệu.

BÀI TẬP VÀ CÂU HỎI

1. Định nghĩa các phép toán của ngôn ngữ đại số quan hệ, cho các ví dụ minh hoạ.
2. Nêu các ứng dụng của ngôn ngữ đại số quan hệ, cho các ví dụ.
3. Trình bày các lệnh định nghĩa dữ liệu của SQL, minh hoạ bằng ví dụ.
4. Trình bày các lệnh thao tác dữ liệu của SQL, cho ví dụ minh hoạ.
5. Cho các quan hệ sau:

r (A B C)

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

s (D E F)

D	E	F
d_1	e_1	f_1
d_2	e_2	f_2
d_3	e_3	f_3

t (A B D)

A	B	D
a_1	b_1	d_1
a_2	b_2	d_2
a_1	b_3	d_3
a_2	b_1	d_2
a_2	b_2	d_1
a_3	b_3	d_3
a_2	b_3	d_3

a) Hãy tính giá trị của các biểu thức đại số quan hệ:

- r*s*t.
- $\Pi_{CF}(\sigma_{C = c2 \wedge F = f2}(r \times s))$

b) Hãy tính kết quả các biểu thức đại số sau:

- (r × s)*t
- Π_{CEQ}(r*s*t)

với:

r (A B C)

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_2

s (D E)

D	E
d_1	e_1
d_2	e_2
d_3	e_3

t(A B D Q)

A	B	D	Q
a_1	b_1	d_1	q_1
a_1	b_2	d_1	q_2
a_2	b_2	d_2	q_3
a_2	b_2	d_3	q_4

c) Hãy tính kết quả của các phép toán đại số quan hệ:

- r1*r2*r3
- $\sigma_{C=1}$((r1 × r2)*r3)

Với:

r1 (A B C)

A	B	C
a_1	b_1	1
a_2	b_2	2

r2 (C D)

C	D
1	d_1
1	d_2
2	d_2

r3 (B D)

B	D
b_1	d_1
b_2	d_2
b_2	d_1

d) Hãy đơn giản và tính kết quả cho các biểu thức đại số sau:

r (A B C)

A	B	C
a_1	b_1	1
a_2	b_2	2
a_3	b_3	3

s (E F G)

E	F	G
e_1	1	1
e_2	2	2
e_3	1	3

t (A E G)

A	E	G
a_1	e_2	11
a_2	e_1	12
a_3	e_2	22
a_3	e_3	23

$\sigma_{B=b1}$(r*t)

Π_{BF}(r*s*t)

6. Hãy đơn giản và tính kết quả của biểu thức đại số quan hệ sau:

- $\Pi_{BE}(\sigma_{C=c1}(r*s*t))$ - $\Pi_{AD}(t) \div \Pi_{D}(s)$

Với:

r (A B C)

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_4	c_4

s (D E F)

D	E	F
d_1	e_1	f_1
d_2	e_2	f_2
d_3	e_3	f_3

t (A D G)

A	D	G
a_1	d_1	11
a_1	d_2	12
a_2	d_2	21
a_2	d_2	22
a_3	d_1	31
a_1	d_3	13
a_3	d_3	33

7. Xét hai quan hệ

r

Chuyến bay	Máy bay
VN83	BO767
VN84	BO777
VN85	A320
VN86	ATR72
VN94	F101

s

Phi công	Máy bay
Th.Tuấn	A320
Ph.Long	BO767
T. Thành	BO777
Q.Anh	ATR72
Th.Tuấn	F101

Trong đó quan hệ r mô tả chuyến bay *cb* sẽ do loại máy bay *mb* nào đó đảm nhiệm. Trong khi đó quan hệ s mô tả phi công *fc* có thể lái máy bay thuộc loại gì.

Dựa trên hai quan hệ trên, bằng một phép toán của ngôn ngữ đại số quan hệ, hãy tạo ra một quan hệ thể hiện toàn bộ các khả năng thực hiện các chuyến bay.

8. Cho cơ sở dữ liệu gồm các quan hệ

Nhânviên (<u>MãNV</u>, HọTên, Ngàysinh, MãPhòng)

Phòng (<u>MãPhòng</u>, TênPhòng, Địađiểm, SốĐT)

Dựán (<u>MãDA</u>, TênDA, Ngânsách)

Thamgia (<u>MãNV, MãDA</u>, Sốgiờthamgia)

Biểu diễn các câu truy vấn sau bằng cả ngôn ngữ SQL và đại số quan hệ:

a) Đưa ra danh sách (HọTên, Ngàysinh) của các nhân viên tham gia dự án có tên là 'Đào tạo từ xa' hoặc tham gia dự án có tên là 'QLTC'.

b) Đưa ra danh sách (Tênphòng, Địa điểm) của phòng có nhân viên mã số 'NV-101' làm việc.

c) Cho biết danh sách (HọTên, Ngày sinh, MãPhòng) của các nhân viên tham gia tất cả các dự án.

9. Cho cơ sở dữ liệu gồm các quan hệ sau:

Nhânviên(#NV, Hoten, Diachi, NgaySinh).

DuAn(#DA, TenDA, ChuDauTu, NganSach).

ThanGia(#NV, #DA, SoGioLamviec).

Hãy biểu diễn các câu hỏi sau bằng SQL và đại số quan hệ nếu có thể:

a) Đưa ra danh sách #NV, Họ Tên, Ngày sinh của những nhân viên làm cho dự án có chủ đầu tư là "VINACO".

b) Đưa ra danh sách bao gồm Tên dự án, chủ đầu tư của các dự án có ngân sách khoảng từ 10.000.000 đến 25.000.000.

10. Cho CSDL bao gồm các quan hệ:

CB(MãCB, TênCB, Tel, Nsinh, Lương, MãPh)

Phòng(MãPh, TênPh, TrưởngPh)

TĐVH(MãCb, Trìnhđộ, Th_gian)

Hãy biểu diễn các yêu cầu sau bằng ngôn ngữ đại số quan hệ nếu có thể:

a) Đưa ra danh sách cán bộ có trình độ "Tiến sĩ" của phòng "kỹ thuật".

b) Thay đổi số điện thoại cho cán bộ có mã là "CB01' với số điện thoại mới là 8694634.

11. Cho cơ sở dữ liệu gồm các quan hệ:

SV(#SV, TênVS, MãL, ĐịaChỉ)

LOP(MãL,TênL, SốSV)

CBGD(#CBGD,TênCb, MônDạy, MãL, SốTiết).

Hãy tìm lời giải đại số quan hệ cho câu hỏi:

- Đưa ra danh sách sinh viên học lớp 'CĐ3A' có địa chỉ ở 'Hà Nội'
- Hãy cho biết một danh sách bao gồm: tên cán bộ, môn dạy, số tiết dạy cho lớp 'TIN2B'.
- Đưa ra danh sách các cán bộ dạy môn 'cơ sở dữ liệu'.

12. Cho CSDL bao gồm các quan hệ:

NV(#NV,TênNV, #Ph, lương) (Quan hệ nhân viên)

Phòng(#Ph, TênPh, SốNV)

NN(#NV, NNgữ, TrìnhĐộ) (Quan hệ ngoại ngữ)

Thực hiện các nhiệm vụ sau:

- Dùng SQL tạo các bảng trên.

- Biểu diễn các yêu cầu sau bằng SQL và đại số quan hệ nếu có thể:

  + Cho biết tên các nhân viên phòng 'Kỹ thuật' biết tiếng Anh trình độ C và lương trên 4.47.

  + Xoá nhân viên với mã là 'NV009'.

13. Cho lược đồ quan hệ sau:

Khách sạn (MãKS, TênKS, Địachỉ).

Phòng (SốP, MãKS, LoạiP, Giá).

Đặtphòng (MãKS, MãKhách, Ngàynhận, Ngàytrả, SốP).

Khách (MãKhách, Họtên, Địachỉ).

Biểu diễn các yêu cầu sau bằng SQL và đại số quan hệ nếu có thể:

1. Đưa ra danh sách Giá và LoạiP của tất cả các phòng tại khách sạn Melia.
2. Liệt kê tất cả các khách đang ở tại khách sạn Melia.
3. Liệt kê tất cả các phòng tại khách sạn Melia, bao gồm cả tên của những người khách đang ở tại phòng nếu phòng đó đang có người ở.
4. Hãy liệt kê các phòng không có người ở tại khách sạn Melia.
5. Hãy cho biết tổng số phòng tại mỗi khách sạn tại London.
6. Hãy viết các câu lệnh tạo các quan hệ nói trên.
7. Tăng giá của tất cả các phòng đơn lên 5%.

14. Cho cơ sở dữ liệu gồm các quan hệ sau:

Kháchhàng (MãKH, Họtên, Sốđiệnthoại, Cơquan).

Nhàchothuê (MãN, Địachỉ, Giáthuê, Tênchủnhà).

Hợpđồng (MãN, MãKH, Ngàybắtđầu, Ngàykếtthúc).

Dùng các câu lệnh SQL tạo lập các bảng trên.

Biểu diễn các yêu cầu sau bằng ngôn ngữ SQL và biểu thức đại số quan hệ (nếu có thể).

a) Đưa ra danh sách {Địachỉ, Tênchủnhà} của những ngôi nhà có giá thuê ít hơn 300.000.

b) Đưa ra danh sách {MãKH, Họtên, Cơquan} của những người đã từng thuê nhà của chủ nhà có tên là Nguyễn Văn A.

c) Đưa ra danh sách các ngôi nhà chưa từng được ai thuê.

d) Hãy đưa ra giá thuê cao nhất trong số các giá thuê của các ngôi nhà đã từng ít nhất một lần được thuê.

15. Cho cơ sở dữ liệu gồm các quan hệ:

Sinhviên(MãSV, Họtên, Ngàysinh, Quêquán).

Mônhọc(MãMH, TênMH, MãGV).

Giảngviên(MãGV, Họtên, Địachỉ, Điệnthoại).

Đăngký(MãMH, MãSV, Kỳhọc, Điểm).

a) Dùng các câu lệnh SQL, tạo các bảng tương ứng với các quan hệ cho ở trên.

b) Biểu diễn các câu truy vấn sau bằng ngôn ngữ SQL và đại số quan hệ.

- Những sinh viên nào học các môn do giảng viên "Lê Tuấn Quang" dạy.
- Chỉ ra tên của tất cả các sinh viên không đăng ký học môn nào.

c) Viết câu lệnh SQL cho các câu truy vấn sau

- Tất cả các sinh viên được 4 điểm trong môn học Comp3311 ở kỳ học '01-S2' sẽ được nâng lên thành 5 điểm.
- Đưa ra họ tên và điểm trung bình của các sinh viên trong học kỳ 'S2-02'

16. Cho cơ sở dữ liệu gồm các quan hệ sau:

Nhânviên (<u>MãNV</u>, Họtên, Địachỉ, Ngàysinh).

Dự án (<u>MãDA</u>, TênDA,Chủđầutư, Ngânsách).

Làmviệc(MãNV, MãDA, Sốgiờlàmviệc).

Dùng các câu lệnh SQL tạo lập các bảng trên.

Biểu diễn các yêu cầu sau bằng ngôn ngữ SQL và biểu thức đại số quan hệ (nếu có thể):

a) Đưa ra danh sách tên và chủ đầu tư của những dự án ngân sách nhiều hơn 10.000.000 và ít hơn 25.000.000.

b) Đưa ra danh sách {MãNV, Họtên, Ngàysinh} của những nhân viên đã từng làm cho các dự án có chủ đầu tư là VINACO.

c) Đưa ra danh sách các nhân viên chưa từng tham gia dự án nào.

d) Hãy đưa ra tổng số giờ mà nhân viên có tên là Nguyễn Văn An đã làm cho các dự án.

17. Cho cơ sở dữ liệu gồm các quan hệ sau:

Nhânviên(#NV,Hoten, Diachi, NgaySinh)

DuAn(#DA,TenDA, ChuDauTu, NganSach)

ThamGia(#NV,#DA, SoGioLamviec)

Yêu cầu:

1. Dùng SQL tạo lập cơ sở dữ liệu trên.

2. Biểu diễn các câu hỏi sau bằng SQL và đại số quan hệ nếu có thể:

a)Đưa ra danh sách #NV, Họ Tên, Ngày sinh của những nhân viên làm cho dự án có chủ đầu tư là "VINACONEX".

b) Đưa ra danh sách bao gồm: Tên dự án, chủ đầu tư của các dự án có ngân sách khoảng từ 10.000.000 đến 25.000.000.

c) Đưa ra tổng số giờ làm việc của Tạ Hiền trong các dự án do Tổng Công ty Sông Đà làm chủ đầu tư.

d) Xoá dự án có tên "White House" khỏi cơ sở dữ liệu.

18. Cho cơ sở dữ liệu gồm các quan hệ:

Nhânviên (<u>MãNV</u>, HọTên, Ngàysinh, MãPhòng)

Phòng (<u>MãPhòng</u>, TênPhòng, Địađiểm, SốĐT)

Dựán (<u>MãDA</u>, TênDA, Ngânsách)

Thamgia (<u>MãNV, MãDA</u>, Sốgiờthamgia)

a) Dùng các câu lệnh SQL, tạo các bảng tương ứng với các quan hệ cho ở trên (thuộc tính được gạch chân là khoá chính của quan hệ).

b) Biểu diễn các câu truy vấn sau bằng cả ngôn ngữ SQL và đại số quan hệ.

- Đưa ra danh sách (HọTên, Ngàysinh) của các nhân viên tham gia dự án có tên là 'Đào tạo từ xa' hoặc tham gia dự án có tên là 'QLTC'.
- Đưa ra danh sách (Tênphòng, Địa điểm) của phòng có nhân viên mã số 'NV-101' làm việc.
- Cho biết danh sách (HọTên, Ngày sinh, MãPhòng) của các nhân viên tham gia tất cả các dự án.

c) Viết câu lệnh SQL cho các câu truy vấn sau:

- Cho biết có bao nhiêu dự án có ngân sách lớn hơn 100.000.000.
- Xóa nhân viên có mã số 'NV-202'.

Chương 5

LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ

Cơ sở dữ liệu (CSDL) là nơi lưu trữ lâu dài các dữ liệu của hệ thống ở bộ nhớ ngoài. CSDL này phải được tổ chức tốt theo hai tiêu chí:

1. *Hợp lý*, nghĩa là phải đủ dùng, không dư thừa.

2. *Tiện lợi khi truy nhập*, nghĩa là các thao tác tìm kiếm, cập nhật, bổ sung và loại bỏ các thông tin phải diễn ra một cách nhanh chóng, thuận tiện.

Nội dung của chương này là giới thiệu một số công cụ thường dùng trong quá trình xây dựng CSDL.

Trước hết, hãy xét đến cơ sở lý thuyết của vấn đề này.

I. PHỤ THUỘC HÀM

Khái niệm về phụ thuộc hàm (trong một quan hệ) là một khái niệm có tầm quan trọng hết sức lớn đối với việc thiết kế các mô hình dữ liệu.

1.1. Một số định nghĩa

Cho một quan hệ r định nghĩa trên lược đồ quan hệ $U = \{A_1, A_2, \ldots, A_n\}$, X và Y là các tập con của U.

Định nghĩa 1

Ta nói: X *xác định hàm* đối với Y trong r, ký hiệu là $X \rightarrow Y$, nếu và chỉ nếu $\forall t, t' \in r$ sao cho $t[X] = t'[X] \Rightarrow t[Y] = t'[Y]$ (hay người ta còn viết $t.X = t'.X \Rightarrow t.Y = t'.Y$).

Nói một cách khác đi $X \rightarrow Y$ khi với $\forall t, t' \in r$ nếu t và t' bằng nhau trên X thì chúng cũng bằng nhau trên Y.

Chú ý: Liên quan đến khái niệm này, chúng ta có một số diễn đạt đồng nghĩa như sau:

- Y phụ thuộc hàm vào X trong r.
- r thoả thuộc hàm $X \rightarrow Y$.
- $X \rightarrow Y$ là phụ thuộc hàm trong r.

Ví dụ: Xét quan hệ S, ta có các phụ thuộc hàm sau:

#S → SNAME, #S → CITY,...

Xét một ví dụ khác, giả sử có quan hệ r như sau:

A	B	C	D
0	0	0	0
1	1	1	1
0	1	0	1

Đối với quan hệ này, ta thấy ngay A→ C, bởi vì với mọi cặp bộ t và t' của r, nếu t[X] = t'[X] thì t[Y] = t'[Y].

Định nghĩa 2

Xét quan hệ r, giả sử r thoả thuộc hàm X → Y và nếu *không tồn tại* một tập con thực sự X' của X sao cho r cũng thoả thuộc hàm X → Y thì ta nói Y *phụ thuộc hàm đầy đủ* vào X trong r.

Người ta dùng ký hiệu FD (Functional Dependency) để chỉ phụ thuộc hàm và FFD (Full Functional Dependency) để chỉ phụ thuộc hàm đầy đủ.

Định lí 2.1

Xét quan hệ r xác định trên tập thuộc tính U và X, Y $\subset$ U, lúc đó:

- X → Y là phụ thuộc hàm trên r khi và chỉ khi X là khoá của quan hệ r(XY).
- X→Y là phụ thuộc hàm đầy đủ trên r khi và chỉ khi X là khoá tối thiểu của quan hệ r(XY).

1.2. Hệ tiên đề cho phụ thuộc hàm

Gọi F là tập tất cả các phụ thuộc hàm đối với lược đồ quan hệ r(U) và X → Y là một phụ thuộc hàm với X,Y $\subseteq$ U, ta nói rằng X → Y được *suy diễn logic* từ F nếu quan hệ r trên r(U) đều thoả các phụ thuộc hàm của F thì cũng thoả X → Y, chẳng hạn F ={A → B, B → C} thì A → C suy ra từ F.

Người ta gọi *bao đóng* của F, ký hiệu là F^+, là tập tất cả các phụ thuộc hàm được suy diễn logic từ F, nếu F = F^+ thì F là họ *đầy đủ* của các phụ thuộc hàm.

Công việc đi tìm F^+ là khó khăn, vấn đề ta quan tâm là: cho phụ thuộc hàm f: X → Y, lúc đó trong điều kiện nào f $\in$ F^+ ?

Để có thể xác định được khoá của một lược đồ quan hệ và các suy diễn logic giữa các phụ thuộc hàm cần thiết phải tính được F^+ từ F. Do vậy đòi hỏi phải có các hệ tiên đề, tập quy tắc của hệ tiên đề được Armstrong đề xuất vào năm 1974, được gọi là *hệ tiên đề Armstrong*.

Hệ tiên đề Armstrong

Gọi r(U) là lược đồ quan hệ với U = {A1, A2,..., An} là tập các thuộc tính; giả sử X,Y,Z ⊆ U, hệ tiên đề Armstrong bao gồm:

A1. *Tính phản xạ*: Nếu Y ⊆ X thì X → Y

A2. *Tính tăng trưởng*: Nếu Z ⊆ U, X → Y thì ZX → ZY.

trong đó ZX = Z ∪ X

A3. *Tính bắc cầu*: Nếu X → Y và Y → Z thì X → Z.

Xét một ví dụ áp dụng: Cho AB → C, C → A, chứng minh BC → ABC

Chứng minh:

Thật vậy, theo giả thiết ta có C → A nên:

Suy ra BC → AB (1) (*tính tăng trưởng, thêm B*), mặt khác lại có:

AB → C (giả thiết), dẫn đến:

AB → ABC (2) (*tính tăng trưởng, thêm* AB).

Vậy BC → ABC (*tính bắc cầu giữa* (1) và (2)).

1.2.1. Bổ đề 1

Hệ tiên đề Armstrong là *đầy đủ*, có nghĩa là nếu F là tập phụ thuộc hàm đúng trên quan hệ r và f: X → Y là một phụ thuộc hàm được suy dẫn từ F nhờ hệ tiên đề Armstrong thì f đúng trên r.

1.2.2. Bổ đề 2

A4. *Tính hợp*: Nếu X → Y và X → Z thì X → YZ.

A5. *Tính tựa bắc cầu*: Nếu X → Y và WY → Z thì XW → Z.

A6. *Tính tách*: Nếu X → Y và Z ⊆ Y thì X → Z.

1.2.3. Bao đóng của một tập thuộc tính

Định nghĩa

Cho F là tập các phụ thuộc hàm trên tập thuộc tính U và X ⊆ U, người ta gọi *bao đóng* của X (*đối với* F), ký hiệu là X^+_F, được định nghĩa như sau:

$$X^+_F = \{A \mid X \to A \in F^+\}$$

(tập hợp tất cả các thuộc tính của U sao cho phụ thuộc hàm X → A có thể suy dẫn từ F nhờ hệ tiên đề Armstrong).

Ghi chú: để tiện và nếu không gây ra sự hiểu lầm thay vì X^+_F, ta chỉ viết đơn giản là X^+.

Tính bao đóng của một tập thuộc tính

Bài toán: Cho U là một tập các thuộc tính và F là tập phụ thuộc hàm trên U, $X \subseteq U$. Hãy tính X^+.

Thuật toán: Vào U, F, X.

Ra X^+.

Tính liên tiếp các $X_0, X_1, X_2, \ldots$, theo quy tắc:

Bước 1: Tam = ϕ

Bước 2: WHILE Tam <> X DO

Tam = X

FOR $\forall f = W \rightarrow Z \in F$ DO

IF $W \subseteq X$ THEN $X = X \cup Z$

Bước 3: RETURN(X)

Ví dụ:

Cho F = {A → D, AB → E, BI → E, CD → I, E → C} và X = AE, sau khi thực hiện thuật toán trên, ta thu được X^+ = ACDEI.

1.2.4. Bổ đề

$X \rightarrow Y$ có thể suy dẫn từ F nhờ hệ tiên đề Armstrong khi và chỉ khi $Y \subseteq X^+$.

1.2.5. Dùng bao đóng của tập thuộc tính để xác định khoá một lược đồ quan hệ

Định lí

Cho U là một tập các thuộc tính, F là tập phụ thuộc hàm trên U và $X \subseteq U$. X là khóa của U dưới F khi và chỉ khi $X_1^+ = U$.

Một ví dụ áp dụng:

Cho U = ABCDEH và F = {C → E, AH → B, B → D, A → D}. Hãy chứng minh rằng AHC là khoá tối thiểu của U. Thực vậy:

- Vì $(AHC)_F^+$ = AHCBDE = U nên AHC là khoá của U.
- Mặt khác ta lại có:
  - $(A)_F^+ = AD \neq U$
  - $(H)_F^+ = H \neq U$
  - $(C)_F^+ = CE \neq U$

- $(AH)_F^+ = AHBD \neq U$
- $(AC)_F^+ = ACDE \neq U$
- $(HC)_F^+ = HCE \neq U$

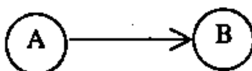
Điều này có nghĩa là bất kỳ tập con nào của AHC đều không là khoá của U, vậy AHC là khoá tối thiểu của U.

1.2.6. Dùng đồ thị của tập phụ thuộc hàm để xác định khoá của một lược đồ quan hệ

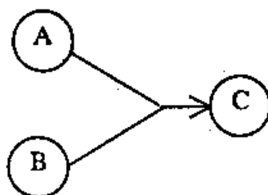
a) Đồ thị của một tập phụ thuộc hàm

Ta có thể biểu diễn một phụ thuộc hàm bằng một đồ thị như sau:

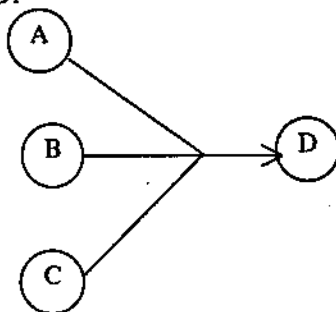
- Phụ thuộc hàm $A \rightarrow B$:



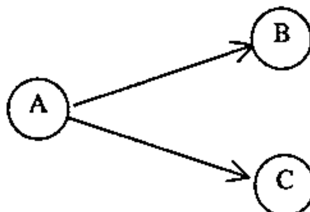
- Phụ thuộc hàm $AB \rightarrow C$



- Phụ thuộc hàm $ABC \rightarrow D$:

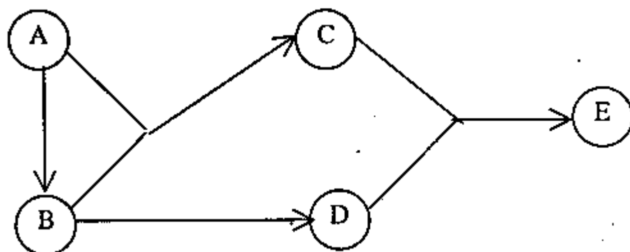


- Phụ thuộc hàm $A \rightarrow BC$:



Đồ thị của một tập phụ thuộc hàm: Cho một tập phụ thuộc hàm F, ta có thể biểu diễn thành một đồ thị với quy tắc: Mỗi thuộc tính là một đỉnh trong đồ thị.

Ví dụ: Cho tập phụ thuộc hàm F = {A → B, AB → C, B → D, CD → E} ta biểu diễn thành một đồ thị sau:



Trong đồ thị, ta định nghĩa:

Đỉnh gốc: là đỉnh mà chỉ có điểm đi của mũi tên.

Ví dụ: Đỉnh A của đồ thị trên.

Đỉnh ngọn: là đỉnh chỉ có điểm đến của mũi tên.

Ví dụ: Đỉnh E của đồ thị trên.

Đỉnh trung gian: là đỉnh vừa có điểm đến, vừa có điểm đi của mũi tên.

Ví dụ: Đỉnh B, C, D của đồ thị trên.

b) Dùng đồ thị của một tập phụ thuộc hàm để xác dịnh khoá của lược đồ quan hệ.

Cho lược đồ quan hệ r(U) và tập phụ thuộc hàm F trên U.

Thuật giải dùng đồ thị để tìm khoá lược đồ quan hệ:

- Các đỉnh gốc của đồ thị *là* thuộc tính khoá.
- Các đỉnh ngọn của đồ thị *không là* thuộc tính khoá
- K là tập các đỉnh gốc:
  - Nếu $K_F^+ = U$ thì K là khoá.
  - Ngược lại, ta thêm lần lượt thêm vào K một số thuộc tính là đỉnh trung gian cho đến khi $K_F^+ = U$.

Trong đồ thị của ví dụ trên, A là khoá của U.

1.2.7. Phủ của tập phụ thuộc hàm

Cho hai tập phụ thuộc hàm F và G trên tập thuộc tính U.

Định nghĩa 1

- Ta nói F *suy ra* G, ký hiệu F |= G, nếu và chỉ nếu $G^+ \subseteq F^+$.
- Ta nói F và G *tương dương*, ký hiệu F ≅ G, nếu và chỉ nếu F |= G và G |= F, khi đó ta nói G là *một phủ* của F và ngược lại.

Định nghĩa 2

Tập phụ thuộc hàm F gọi là *tối thiểu* nếu:

1. $\forall f \in F \Rightarrow f = X \rightarrow A$ (vế phải chỉ có 1 thuộc tính)
2. Không tồn tại $f = X \rightarrow A \in F$ và $Z \subset X$ thoả $F^+ = (F \backslash \{F\} \cup \{Z \rightarrow A\})^+$
3. Không tồn tại $f = X \rightarrow A \in F$ sao cho $F^+ = (F \backslash \{f\})^+$

Định nghĩa 3

Tập phụ thuộc hàm F, tập phụ thuộc hàm G gọi là *phủ tối thiểu* của F khi và chỉ khi G'là một phủ của F và G là tối thiểu.

Thuật giải tìm phủ tối thiểu của tập phụ thuộc hàm

Vào: Tập phụ thuộc hàm F trên U.

Ra: G là phủ tối thiểu của F.

- *Bước1*: $G = \phi$. Tách tất cả các phụ thuộc hàm của f thành phụ thuộc hàm mà vế phải chỉ có 1 thuộc tính:

  FOR $\forall f \in F$, $f = X \rightarrow$ DO $G = G \cup \{X \rightarrow A, A \in Y\}$

- *Bước 2*: Loại bỏ những phụ thuộc hàm không đầy đủ:

WHILE $\exists Z \subset X, Z \neq X, G \cong G \backslash \{f\} \cup \{Z \rightarrow A\}$ DO $G = G \backslash \{f\} \cup \{Z \rightarrow A\}$

- *Bước 3*: Loại bỏ những phụ thuộc hàm dư thừa:

  FOR $\forall f \in G$ DO

  IF $G \backslash \{f\} \cong G$ THEN $G = G \backslash \{f\}$

- *Bước 4*: RETURN(G).

Ta có thể diễn giải lại thuật giải trên như sau:

- *Bước 1:* Tất cả các phụ thuộc hàm của F thành phụ thuộc hàm mà vế phải chỉ có một thuộc tính, chẳng hạn:

$AB \rightarrow CD$ được tách thành $AB \rightarrow C$ và $AB \rightarrow C$ (luật tách)

- *Bước 2:* Loại bỏ những phụ thuộc hàm không đầy đủ. Khi loại bỏ, ta phân biệt hai loại phụ thuộc hàm không đầy đủ sau:

  Loại 1: Phụ thuộc hàm có vế phải là tập con của vế trái (loại $AB \rightarrow B$).

  Loại 2: Hai phụ thuộc hàm có vế phải giống nhau, nếu vế trái của phụ thuộc hàm này chứa vế trái của phụ thuộc hàm kia thì ta loại ra khỏi F.

Chẳng hạn: nếu có $ABC \rightarrow D$ và $BC \rightarrow D$ thì ta loại $ABC \rightarrow D$ khỏi F.

- *Bước 3:* Loại bỏ những phụ thuộc hàm dư thừa:

Giả sử hai phụ thuộc hàm có vế phải giống nhau: $f_1 = X \rightarrow A$ và

$f_2 = Y \rightarrow A$, lúc đó nếu có $A \in X^+_{F\setminus\{0\}}$ thì loại f_1 ra khỏi F.

Ví dụ sau đây minh hoạ cho quy trình trên:

Cho F = {AB →C, C→A, BC→ D, ACD → B, D → EG, BE → C, CG→ BD, CE → AG}

Thực hiện thuật toán ta có:

Sau bước 1: G = {AB → C, C→ A, BC→ D, ACD → B, D → E, D → G, BE → C, CG → B, CG → D, CE → A, CE → G }

Sau bước 2: G = {AB →C, C→A, BC→ D, CD → B, D → E, D → G, BE → C, CG → B, CG → D, CE → G }

Sau bước 3: G = {AB →C, C→A, BC→ D, D → E, D → G, BE → C, CG → B, CE → G }

Sau bước 4: G là phủ tối thiểu của F.

II. TÁCH MỘT QUAN HỆ

Như ta đã biết mô hình quan hệ do Codd đề xuất năm 1970, có những ưu điểm vượt trội so với các mô hình dữ liệu trước đó:

- *Đơn giản*: các dữ liệu được biểu diễn dưới một dạng duy nhất, là các bảng giá trị, khá tự nhiên và dễ hiểu đối với mọi người sử dụng.
- *Chặt chẽ*: các khái niệm được hình thức hoá cao, cho phép sử dụng các công cụ toán học, các thuật toán.
- *Trừu tượng hoá cao:* mô hình chỉ dừng ở mức quan niệm, nghĩa là độc lập với mức vật lý, với sự cài đặt, với các thiết bị. Nhờ đó làm tăng thêm tính độc lập giữa dữ liệu và chương trình.
- *Cung cấp các ngôn ngữ truy nhập dữ liệu ở mức cao* (như SQL,...): nhờ đó dễ sử dụng và trở thành chuẩn.

Tuy vậy, khi thiết kế một cơ sở dữ liệu quan hệ thường đòi hỏi phải chọn các lược đồ quan hệ. Việc chọn tập các lược đồ này có thể tốt hơn hay xấu hơn tập các lược đồ khác dựa trên một số tiêu chuẩn nào đó. Trọng tâm của việc thiết kế các lược đồ cơ sở dữ liệu là ta tổ chức bao nhiêu lược đồ và mỗi lược đồ có những thuộc tính nào để bảo đảm các tính chất sau:

Không trùng lắp dữ liệu: Trong một quan hệ, giá trị của một thuộc tính nào đó chiếm dung lượng bộ nhớ lớn không được lặp lại nhiều lần.

Ví dụ: Xét lược đồ quan hệ:

BANHANG(M_HANG, K_HANG, DIACHIKH, SOLUONG, DONGIA)

Dễ dàng thấy rằng, mỗi khách hàng khi mua một mặt hàng nào đó thì tên, địa chỉ của khách hàng lại xuất hiện một lần nữa. Tên của khách hàng và địa chỉ khách hàng được lặp lại nhiều lần trong quan hệ này nên đây là một lược đồ quan hệ không tốt, bị trùng lắp dữ liệu.

Nhất quán dữ liệu: Trong một lược đồ quan hệ xác định được nhiều phụ thuộc hàm, tất cả các quan hệ xác định trên lược đồ quan hệ phải thoả các phụ thuộc hàm trên lược đồ ấy.

Ví dụ: Xét lược đồ quan hệ **BANHANG** ở trên, mỗi khách hàng khi mua một mặt hàng nào đó thì ta phải nhập tên, địa chỉ của khách hàng này một lần nữa. Tên, địa chỉ khách hàng lần nhập sau có thể khác lần nhập trước như: Trùng tên nhưng khác địa chỉ hay trùng địa chỉ nhưng khác tên hay khác cả tên lẫn địa chỉ dẫn đến việc là máy coi hai người này là khác nhau. Hơn nữa, khi một khách hàng đổi địa chỉ, ta không thể tìm đổi hết các địa chỉ trong các bộ của quan hệ. Đây là một lược đồ quan hệ không nhất quán dữ liệu.

Không gây dị thường khi thêm bộ: Một khách hàng khi chưa mua hàng, thì không thể bổ sung thông tin của ông ta vào CSDL ngay được.

....

Để tạo một cơ sở dữ liệu tốt hơn, nghĩa là không trùng lắp thông tin, nhất quán dữ liệu, ta phải tách một lược đồ quan hệ thành nhiều lược đồ con.

2.1. Tách một lược đồ quan hệ

Phép tách một lược đồ quan hệ $U = A_1A_2...A_n$ là việc thay thế lược đồ quan hệ U bằng một tập lược đồ $U_1, U_2,..., U_n$ trong đó $U_i \subseteq U$, i = 1.. n.

$U = U_1 \cup U_2 \cup ... \cup U_n$ và $U_i \neq U_j$ với $i \neq j$.

Ví dụ: Với lược đồ quan hệ: r(M_HANG, MAKH, TENKH, DCKH, SOLUONG, DONGIA) ta có phụ thuộc hàm:

M_HANG, MAKH → SOLUONG, DONGIA

MAKH → TENKH, DCKH

Ta có thể tách lược đồ quan hệ **r** thành hai lược đồ như sau:

r_1(M_HANG, MAKH, SOLUONG, DONGIA), r_2(MAKH, TENKH, DCKH)

Việc phân rã một lược đồ phụ thuộc tập phụ thuộc hàm xác định trên lược đồ ấy.

2.1.1. Phép tách bảo toàn thông tin

Cho lược đồ quan hệ r và F là tập phụ thuộc hàm xác định trên r.

Phép tách lược đồ r thành các lược đồ con $r_1, r_2, \ldots, r_n$ dựa trên tập F gọi là phép tách **bảo toàn thông tin** nếu: Với mọi quan hệ r trên r ta đều có r là phép kết nối tự nhiên của các phép chiếu của r lên các r_i:

$$\forall r(r) \Rightarrow r = \prod_{R1}(r)^{*}\prod_{R2}(r)^{*}\ldots^{*}\prod_{Rn}(r)$$

2.1.2. Kiểm tra phép tách bảo toàn thông tin

Liệu một phép tách lược đồ r thành các lược đồ con $r_1, r_2, \ldots, r_n$ dựa trên tập F bảo toàn thông tin hay không, được kiểm tra qua phương pháp sau đây gọi là *phương pháp tableau*:

Thuật toán: Kiểm tra phép tách bảo toàn thông tin.

Vào: Lược đồ $r = A_1A_2\ldots A_n$, tập F các phụ thuộc hàm trên r và phép tách $\rho = \{r_1, r_2, \ldots, r_k\}$

Ra: ρ có bảo toàn thông tin hay không?

Bước 1: Lập một bảng (Tableau) gồm n cột và k hàng, cột thứ j là thuộc tính A_j của r, hàng thứ i tương ứng lược đồ con r_i, tại vị trí hàng i cột j ta điền ký hiệu a_j nếu $A_j \in r_i$, điền ký hiệu b_{ij} nếu $A_j \notin r_i$.

Bước 2: Áp dụng quy trình thay thế đuổi trên bảng trên:

Với $f = X \rightarrow Y \in F$ thì: Xét các hàng có giá trị bằng nhau trên các thuộc tính của X thì làm bằng nhau các giá trị trên các thuộc tính của Y theo nguyên tắc: nếu trên các hàng bằng nhau ấy có giá trị a_j thì thay các giá trị bởi a_j, ngược lại thay các b_{ij} bởi một b_{ij} tuỳ ý. Áp dụng việc thay thế này cho đến khi không còn tạo ra được bảng mới nữa hay có một hàng chỉ toàn các giá trị a_j.

Bước 3: Nếu trong bảng sau cùng tồn tại một hàng toàn các giá trị a_j thì P bảo toàn thông tin, ngược lại thì P không bảo toàn thông tin.

Xét một vài ví dụ minh hoạ cho thuật toán trên:

Ví dụ 1: Cho r = ABCD, F = {A → B, AC → D} và phân rã

ρ = {(AB), (ACD)} ta có:

$T_1(R)$(A	B	C	D)
a_1	a_2	b_{13}	b_{14}
a_1	b_{22}	a_3	a_4

Với $A \to B$ ta có

$T_2(R)$(A	B	C	D)
a_1	a_2	b_{13}	b_{14}
a_1	a_1	a_3	a_4

Chúng ta thấy rằng ở bảng $T_2(r)$ xuất hiện hàng thứ hai toàn a_j nên phép tách ρ bảo toàn thông tin.

Ví dụ 2: Cho r = ABCD, F = {A → B, AC → D} và phân rã

ρ = {(AC), (BCD)} ta có:

$T_1(R)$(A	B	C	D)
a_1	b_{12}	a_3	b_{14}
b_{12}	a_2	a_3	a_4

Với $A \to B$ ta có

$T_2(R)$(A	B	C	D)
a_1	b_{12}	a_3	b_{14}
b_{21}	a_2	a_3	a_4

Với $AC \to D$ ta có

$T_3(R)$(A	B	C	D)
a_1	b_{12}	a_3	b_{14}
b_{21}	a_2	a_3	a_4

Ta thấy ở bảng $T_1(R) \equiv T_2(R) \equiv T_3(R)$ không xuất hiện hàng nào toàn a_j nên phép tách ρ không bảo toàn thông tin.

Trong trường hợp $\rho = \{r_1, r_2\}$, nghĩa là phép tách chỉ có 2 lược đồ con. Để kiểm tra phép tách bảo toàn thông tin, ta sử dụng định lí sau:

Định lí

Phép phân rã r thành $\rho = \{r_1(U_1), r_2(U_2)\}$ là bảo toàn thông tin

$$\Leftrightarrow U_1 \cap U_1 \to U_1 \setminus U_2 \text{ hay } U_1 \cap U_2 \to U_2 \setminus U_1$$

Xét lại hai ví dụ trên ta có:

r = ABCD, F = {A → B, AC → D} và phân rã ρ = {(AB), (ACD)}, trong trường hợp này:

Ký hiệu $U_1 = AB$ và $U_2 = ACD \Rightarrow U_1 \cap U_2 = A \to U_1 \setminus U_2 = B$. Vậy theo định lí vừa xét ρ bảo toàn thông tin.

2.2. Chuẩn hoá lược đồ quan hệ

Khi thiết kế một lược đồ quan hệ phải tuân theo một số nguyên tắc để khi thao tác trên cơ sở dữ liệu không dẫn đến sự dị thường dữ liệu. Công việc thiết kế dữ liệu theo một dạng chuẩn nào đó gọi là chuẩn hóa dữ liệu.

Lý thuyết cơ sở dữ liệu hiện nay đã đưa ra nhiều dạng chuẩn, tuy nhiên, xét thấy rằng dữ liệu trong thực tế không đến nỗi quá phức tạp, trong quá trình này chúng tôi chỉ đưa ra ba dạng chuẩn nhất là dạng chuẩn 1, dạng chuẩn 2 và dạng chuẩn 3.

2.2.1. Các dạng chuẩn trong lược đồ quan hệ

Do việc cập nhật dữ liệu (qua phép chèn, loại bỏ, sửa đổi) gây nên những dị thường, cho nên các quan hệ nhất thiết phải được biến đổi thành các dạng phù hợp, quá trình đó được gọi là quá trình chuẩn hoá. Quan hệ được chuẩn hoá là quan hệ trong đó mỗi miền của một thuộc tính chỉ chứa những giá trị nguyên tố, tức là không phân nhỏ được nữa và do đó mỗi giá trị trong quan hệ cũng là nguyên tố.

Quan hệ có chứa các miền trị là không nguyên tố gọi là quan hệ không chuẩn hoá. Một quan hệ được chuẩn hoá có thể tách thành một hoặc nhiều quan hệ chuẩn hoá khác và không làm mất thông tin. Giới hạn trong việc xem xét các lược đồ quan hệ với các phụ thuộc hàm, ta định nghĩa dưới đây các dạng chuẩn 1NF, 2NF, 3NF.

2.2.2. Một số định nghĩa

Thuộc tính khoá: Một thuộc tính trong lược đồ quan hệ r(U) được gọi là thuộc tính khoá nếu nó là một thành phần của một khoá nào đó của r. Ngược lại được gọi là thuộc tính không khoá.

Phụ thuộc hàm đầy đủ: Cho lược đồ quan hệ r (U), U là tập thuộc tính, X và Y là hai tập con khác nhau của U. Y là phụ thuộc hàm đầy đủ vào X nếu Y là phụ thuộc hàm vào X nhưng không phụ thuộc hàm vào bất kỳ tập hợp con thực sự nào của X.

Phụ thuộc bắc cầu: Cho lược đồ quan hệ r (U), U là tập thuộc tính, X là tập con của U. A là một thuộc tính thuộc U. A được gọi là phụ thuộc bắc cầu vào X trên r nếu tồn tại một tập con Y của U sao cho X → Y, Y → A nhưng Y → X với A ∉ XY.

2.2.3. Dạng chuẩn 1 (First Normal Form - 1NF)

Định nghĩa: Một lược đồ quan hệ r(U) được gọi là ở dạng chuẩn 1 (1NF) nếu và chỉ nếu toàn bộ các miền của các thuộc tính có mặt trong r đều chỉ chứa các giá trị nguyên tố.

Ví dụ 1: Xét lược đồ: HOADON (MAHANG, MAKH, SOLUONG, DONGIA, THANHTIEN) không là dạng chuẩn 1 vì có giá trị thuộc tính THANHTIEN=SOLUONG×DONGIA.

Ví dụ 2: Xét quan hệ S(S#, PRO), trong trường hợp này (hình 5.1) ta thấy ngay thuộc tính PRO có các giá trị không nguyên tố, nên lược đồ này không thể là dạng 1NF.

S#	PRO	
	P#	QTY
s1	100	1
	200	2
	300	1
2	100	4
	200	2
3	400	5
	500	1

Hình 5.1

2.2.4. Dạng chuẩn 2

(Second Normal Form - 2NF)

Trước khi định nghĩa dạng chuẩn 2, ta xét một ví dụ: Cho CSDL gồm hai quan hệ:

THI	(Môn thi,	Giáo viên)
	3	Đinh
	4	Dậu
	5	Bính

SINHVIEN	(MONTHI,	MSSV,	TEN,	TUOI,	DIACHI,	DIEM)
	3	11	LAN	20	X	8
	3	12	HA	21	Y	6
	4	11	LAN	20	X	7
	4	12	HA	21	Y	6
	5	11	LAN	20	X	7
	5	13	TU	22	Z	2

Trong lược đồ THI thì **MONTHI** là **khoá**, trong lược đồ SINHVIEN thì **MONTHI, MSSV** (Mã số sinh viên) là **khoá**.

Trong quan hệ SINHVIEN thì các thuộc tính MONTHI, MSSV, DIEM xác định kết quả học tập của sinh viên còn MSSV, TEN, TUOI, DIACHI xác định đối tượng là sinh viên, nghĩa là trên SINHVIEN ta có hai phụ thuộc hàm: MONTHI, MSSV → DIEM và MSSV → TEN, TUOI, DIACHI, nghĩa là thuộc tính không khoá. TEN, TUOI, DIACHI phụ thuộc không đầy đủ vào khoá của quan hệ.

Trong quan hệ này, ta nhận thấy: Việc lưu trữ thông tin một sinh viên như LAN, nếu học n môn thì lặp lại n lần tên, tuổi, địa chỉ, ta thấy quá dư thừa! Từ việc dư thừa này dẫn đến những sai phạm khi thao tác dữ liệu:

Bổ sung: Khi thêm một sinh viên học một môn nào đó, ta không biết sinh viên này đã có chưa nên một lần nữa lại nhập tên, tuổi, địa chỉ của sinh viên này. Trong quá trình thêm này có thể nhập sai tuổi chẳng hạn, dẫn đến không nhất quán dữ liệu: một sinh viên có hai tuổi khác nhau.

Loại bỏ: Khi một sinh viên không còn học một môn học nào đó, ta phải xoá sinh viên này khỏi quan hệ. Nếu trong quan hệ chỉ còn có một bộ chứa sinh viên này mà ta xoá đi, dẫn đến mất hẳn thông tin về sinh viên này trong hệ thống.

Sửa dữ liệu: Khi cần điều chỉnh một số thông tin nào đó của một sinh viên, ta phải sửa tất cả các bộ của sinh viên này. Trong quá trình sửa này có thể sót một bộ nào đó dẫn đến không nhất quán dữ liệu: một sinh viên có hai thông tin khác nhau.

Để tổ chức dữ liệu tốt hơn, ta nên tách lược đồ SINHVIEN thành hai lược đồ như sau: SINHVIEN(**MSSV**, TEN, TUOI, DIACHI) và DIEMTHI (**MSSV**, **MONTHI**, DIEM).

Định nghĩa: Một lược đồ quan hệ r(U) được gọi là ở dạng chuẩn 2 nếu nó thoả mãn điều kiện:

- Là dạng chuẩn 1.
- Mọi thuộc tính không khoá của r phụ thuộc đầy đủ vào khoá chính.

Nói cách khác LĐQH là dạng 2NF nếu không có tập con thực sự của khoá kéo theo thuộc tính không khoá.

Ví dụ: Lược đồ SINHVIEN (**MONTHI**, **MSSV**, TEN, TUOI, DIACHI, DIEM) không là dạng chuẩn 2.

Trong khi đó:

SINHVIEN (**MSSV**, TEN, TUOI, DIACHI) và DIEMTHI (**MSSV**, **MONTHI**, DIEM) là dạng chuẩn 2.

2.2.5. Dạng chuẩn 3 (Third Normal Form - 3NF)

Định nghĩa: Một lược đồ quan hệ r được gọi là ở dạng chuẩn 3 nếu nó thoả mãn hai điều kiện:

- Là dạng chuẩn 2.
- Mọi thuộc tính không khoá của r là không phụ thuộc bắc cầu vào khoá chính.

Ví dụ 1: Cho lược đồ quan hệ: r = SAIP với các phụ thuộc hàm SI $\rightarrow$ P và S$\rightarrow$ A. Ta có SI là khoá của r, A là thuộc tính không khoá: SI $\rightarrow$ S và S $\rightarrow$ A, nhưng không tồn tại A $\rightarrow$ SI với A $\not\in$ SI. Vậy r không là dạng chuẩn 3.

Ví dụ 2: Cho lược đồ quan hệ: r = CSZD với các phụ thuộc hàm:

CD $\rightarrow$ Z và SZ $\rightarrow$ C. Ta có trong lược đồ này mọi thuộc tính đều là thuộc tính khoá nên nó là dạng chuẩn 3.

2.2.6. Dạng chuẩn một lược đồ cơ sở dữ liệu

Một lược đồ cơ sở dữ liệu (một tập hợp nhiều lược đồ quan hệ) gọi ở dạng chuẩn i (i = 1, 2, 3) nếu mọi lược đồ con của nó đều ở dạng chuẩn i.

2.2.7. Chuẩn hoá một lược đồ cơ sở dữ liệu

Trong những tiết trước ta đã định nghĩa phép tách một lược đồ quan hệ thành nhiều lược đồ con bảo toàn thông tin. Sau đây, ta xét một cách tách một lược đồ quan hệ thành nhiều lược đồ con bảo toàn thông tin dạng chuẩn 3.

Thuật toán : Tách một lược đồ thành 3NF.

- Vào: Lược đồ quan hệ r(U), tập các phụ thuộc hàm F, không làm mất tính tổng quát, giả sử rằng nó là phủ tối thiểu.
- Ra: Phép tách không mất mát thông tin trên r(U), bảo toàn thông tin cho mỗi lược đồ con đều ở 3NF.

Phương pháp:

i. Loại bỏ tất cả các thuộc tính của r nếu các thuộc tính đó không liên quan đến một phụ thuộc hàm nào của F, hoặc vế trái, hoặc vế phải.

ii. Nếu có một phụ thuộc hàm nào của F mà liên quan tới tất cả các thuộc tính của r thì kết quả ra chính là r.

iii. Ngoài ra, phép tách r đưa ra các lược đồ con gồm các thuộc tính XA cho phụ thuộc hàm $(X \rightarrow A) \in F$, tuy nhiên nếu $X \rightarrow A1$, $X \rightarrow A2, \ldots, X \rightarrow An$ thì thay thế tập thuộc tính XA1A2...An cho XAi (i=1,..., n).

Chú ý: Tại mỗi bước kiểm tra lược đồ r, nếu mỗi thuộc tính không khoá không phụ thuộc bắc cầu vào khoá chính, khi đó r sẽ ở 3NF, ngược lại cần áp dụng bước (iii) để tách tiếp.

Ví dụ: Cho lược đồ quan hệ r(CTHRSG) với tập phụ thuộc hàm tối thiểu $\{C \rightarrow T, HR \rightarrow C, CS \rightarrow G$ và $HS \rightarrow R\}$ theo thuật toán vừa xét sẽ thu được tập lược đồ ở dạng chuẩn 3.

BÀI TẬP VÀ CÂU HỎI

1. Định nghĩa phụ thuộc hàm và các khái niệm liên quan.
2. Phát biểu tiên đề Armstrong và các hệ quả.
3. Định nghĩa bao đóng của một tập thuộc tính.

4. Định nghĩa phủ của một tập phụ thuộc hàm. Phủ tối thiểu.

5. Cho lược đồ quan hệ r và tập phụ thuộc hàm:

$$F = \{ab\to e,\ ag\to i,\ be\to i,\ e\to g,\ gi\to h\}$$

Hãy chứng minh rằng $ab\to gh$.

6. Cho lược đồ quan hệ r và tập phụ thuộc hàm:

$$F = \{ab\to c,\ b\to d,\ dc\to e,\ ce\to gh,\ g\to a\}$$

Hãy chứng minh rằng $ab\to e$, $ab\to g$.

7. Cho lược đồ quan hệ S = <U,F> với U={A,B,C,D} và F={a→b, a→c}. Hãy tìm các phụ thuộc hàm suy được từ các quy tắc của phụ thuộc hàm trong các ràng buộc sau:

- a→d.
- c→d
- ab→b.
- bc→a.
- a→bc

8. Xét một quan hệ:

Đào_tạo

Mã	Họ và tên	Ngành học	Môn học	Điểm
Ch1	Trần An	Viễn thông	Cáp quang	8
			Cao tần	7
			Mạch	9
Ch2	Hà Giao	Hoá	Hoá lí	6
			Vô cơ	8
Ch3	Tạ Hiền	CNTT	CSDL	9

Quan hệ trên có phải ở dạng chuẩn 1 không ? Nếu không hãy đưa nó về dạng chuẩn 1.

a) Cho lược đồ quan hệ r và tập các phụ thuộc F = {A →D, AB →DE, CE→G, E→H} xác định trên r. Tính AB_F^+.

b) Cho lược đồ quan hệ r và tập các phụ thuộc hàm F = {A →D, AB →E, BI→E, CD→I, E → C } xác định trên r. Tính AB_F^+.

c) Cho lược đồ quan hệ r và tập các phụ thuộc hàm F = {AB →E, AG → I, BE→I, E → G, GI → H} xác định trên r. Chứng minh rằng $AB \to GH \in F^+$.

d) Cho lược đồ quan hệ r và tập các phụ thuộc hàm F = {A →D, AB → E, BI→E, CD → I, E → C} xác định trên r. Phụ thuộc hàm f: AC → EI xác định trên r có thuộc F^+ hay không?

e) Cho lược đồ quan hệ r và tập các phụ thuộc hàm F = {AB →C, B → D, CD→E, CE → GH, G → A} xác định trên r. Tính AB_F^+. Phụ thuộc hàm f: BG →C xác định trên r có thuộc F^+ hay không?

9. Cho lược đồ quan hệ r và tập các phụ thuộc hàm F = {AB →C, B → D, CD→E, CE →GH, G →A} xác định trên r. Chứng minh rằng AB →E ∈ F^+ và AB → G ∈ F^+.

10. a) Cho lược đồ quan hệ r = ABCD và hai tập các phụ thuộc hàm F = {A → B, A → C, B → A, B → C, C → A}, G = {A → B, B → C}. Chứng minh rằng F ≅ G

b) Cho lược đồ quan hệ r = ABCDE và hai tập các phụ thuộc hàm F = {A→BC, A → D, CD → E}, G = {A → BCE, A → ABD, CD →E}. Chứng minh rằng F ≅ G.

11. Cho tập các phụ thuộc hàm F {A → C, AB →C, C → DI, BC → AB, EI → C} xác định trên r. Tìm một phủ tối thiểu của F.

12. Cho lược đồ quan hệ r = ABC và quan hệ r trên r:

r(A	B	C)
a1	b1	c1
a1	b2	c1
a2	b1	c1

Xét phép tách r thành r_1 và r_2 như sau có bảo toàn thông tin hay không?

r_1(A	C)
a1	c1
a2	c1

r_2(B	C)
b1	c1
b2	c1

13. Cho lược đồ quan hệ r = ABCDE và tập các phụ thuộc hàm F = {A → C, B → C, C → D, DE → C, CE → A}. Xét phép tách r thành các lược đồ con sau: r1 = AD, r2 = AB, r3 = BE, r4 = CDE, r5 = AE. Phép tách này có bảo toàn thông tin hay không?

14. Cho lược đồ quan hệ r = ABCDE và tập các phụ thuộc hàm F = {AB → C, C→ B, ABD → E, F → A}. Xét phép tách r thành các lược đồ con sau: r1 = BC, r2 = AC, r3 = ABDE, r4 = ABDF. Phép tách này có bảo toàn thông tin hay không ?

15. Cho lược đồ quan hệ r = ABCDE và tập các phụ thuộc hàm F = {AB→ C, C → E, C → D, AB → E}. Xét phép tách r thành các lược đồ con sau: r1 = ABC, r2 = AD, r3 = DE. Phép tách này có bảo toàn thông tin hay không ?

16. Cho lược đồ quan hệ r(STUDENT, NAME, BIRTHDAY, AGE, ADVISOR, DEPARTMENT, SEMETER, COURSE, GRADE), và tập các phụ thuộc hàm:

F = {STUDENT → NAME, BIRTHDAY, AGE, ADVISOR, DEPARTMENT,

BIRTHAY → AGE,

ADVISOR → DEPARTMENT}

- Tìm 1 khoá của r dựa vào F.
- Tìm 1 phân rã bảo toàn thông tin của r đối với F.

17. Cho lược đồ quan hệ r(Broker, Office, Investor, Stock, Quantily, Dividend) và các phụ thuộc hàm: F = {S → D, I → B, IS → Q, B → O.

- Tìm một khoá của r dựa vào F.
- Tìm một phân rã bảo toàn thông tin của r đối với F.

18. Cho lược đồ quan hệ r(Ship name, Voyage identifer, Type of shift, Cargo carried one ship on one voyage, Port, Day), và tập các phụ thuộc hàm F = {S → T, V →, SD → PV}.

- Tìm 1 khoá của r dựa vào F.
- Tìm 1 phân rã bảo toàn thông tin của r đối với F.

19. Cho lược đồ quan hệ r(B<u>C</u>DFLIMG) và các tập phụ thuộc hàm F = {C → IBDKF, D →, K → F}, với C là khoá cho trước. Chứng minh r ở dạng chuẩn 3 đối Với F.

20. Cho lược đồ quan hệ r(ABCDEGH) và tập phụ thuộc hàm trên r: F={ABC → D, AB → E, BC → DC, C → ED, CE → H, DC → G, CH → G, AD → H}.

- Tìm 1 phủ tối thiểu của F
- Tìm 1 khoá của r dựa vào phủ tối thiểu của F.
- Tìm 1 phân rã của r dựa trên phủ tối thiểu của F có dạng chuẩn 3 và bảo toàn thông tin.

Phần 2. HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL

Chương 6

TỔNG QUAN VỀ SQL SERVER 2000

Như chúng ta đã biết ở trên, mô hình cơ sở dữ liệu (CSDL) quan hệ do E.F. Codd đưa ra vào đầu thập kỷ 70, từ đó đến nay nó liên tục phát triển và trở thành mô hình CSDL phổ biến bậc nhất. Mô hình quan hệ gồm các thành phần sau:

- Tập hợp các đối tượng và/hoặc, các mối quan hệ.
- Tập hợp các xử lý tác động tới các quan hệ.
- Ràng buộc dữ liệu đảm bảo tính chính xác và nhất quán của dữ liệu.

SQL là viết tắt của cụm từ Structured Query Language, đó là một loại ngôn ngữ dùng để truy cập các cơ sở dữ liệu được thiết lập theo mô hình quan hệ.

Ưu điểm của SQL :

- Gần gũi với ngôn ngữ tự nhiên (tiếng Anh).
- Ngắn gọn, sáng sủa trong chương trình.

Do tính đơn giản, sáng sủa của SQL nên phần lớn các ngôn ngữ lập trình đều được tăng cường nhờ được nhúng ngôn ngữ SQL.

Ngôn ngữ SQL được IBM sử dụng lần đầu tiên trong hệ quản trị CSDL System R vào giữa những năm 70, hệ ngôn ngữ SQL đầu tiên được IBM công bố vào tháng 11 năm 1976. Năm 1979, tập đoàn ORACLE giới thiệu thương phẩm đầu tiên của SQL, SQL cũng được cài đặt trong các hệ quản trị CSDL như DB2 của IBM và SQL/DS.

Ngày nay, SQL được sử dụng rộng rãi và được xem là ngôn ngữ chuẩn để truy cập CSDL quan hệ. Năm 1989, Viện tiêu chuẩn quốc gia Hoa Kỳ (ANSI) công nhận SQL là ngôn ngữ chuẩn để truy cập CSDL quan hệ trong văn bản ANSI SQL89. Cũng trong năm 1989 này, tổ chức tiêu chuẩn quốc tế (ISO) công nhận SQL là ngôn ngữ chuẩn để truy cập CSDL quan hệ trong văn bản ISO 9075-1989. Tất cả các hệ quản trị CSDL lớn trên thế giới cho phép truy cập bằng SQL và hầu hết theo chuẩn ANSI.

Chính bởi tầm quan trọng đó của SQL, trong phần tiếp theo, chúng tôi giới thiệu những nét chính yếu nhất về hệ quản trị cơ sở dữ liệu SQL Server 2000 (còn gọi là phiên bản 8.0) cũng như những đặc tính mới của

phiên bản này và một số kiến thức về ngôn ngữ Transact-SQL (hay còn gọi là T-SQL).

I. SQL SERVER 2000

SQL Server 2000 là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System, viết tắt là RDBMS) sử dụng Transact-SQL để trao đổi dữ liệu giữa Client computer và SQL Server computer. Một RDBMS bao gồm databases (cơ sở dữ liệu), database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS.

SQL Server 2000 được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến TeraByte và có thể phục vụ cùng lúc cho hàng ngàn người sử dụng. SQL Server 2000 có thể kết hợp với các server khác như Microsoft Internet Information Server (IIS),

E-Commerce Server, Proxy Server,....

SQL Server có 7 phiên bản:

- Enterprise : Chứa đầy đủ các đặc trưng của SQL Server và có thể chạy tốt trên hệ thống lên đến 32 CPU và 64 GB RAM. Thêm vào đó nó có các dịch vụ giúp cho việc phân tích dữ liệu (Analysis Services) rất hiệu quả.
- Standard: Rất thích hợp cho các công ty vừa và nhỏ vì giá thành rẻ hơn nhiều so với Enterprise Edition, nhưng lại bị giới hạn một số chức năng cao cấp khác, phiên bản này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.
- Personal: Được tối ưu hóa để chạy trên PC nên có thể cài đặt trên hầu hết các phiên bản Windows kể cả Windows 98.
- Developer: Có đầy đủ các tính năng của Enterprise Edition nhưng giới hạn số lượng người kết nối vào Server cùng một lúc, ... Phiên bản này có thể cài trên Windows 2000 Professional hay Win NT Workstation.
- Desktop Engine (MSDE): Đây chỉ là một chương trình nguồn gốc chạy trên desktop và không có giao diện. Thích hợp cho việc triển khai ứng dụng ở máy Client. Kích thuớc Database bị giới hạn khoảng 2 GB.
- Win CE : Dùng cho các ứng dụng chạy trên Windows CE.
- Trial: Có các tính năng của Enterprise Edition, Download free, nhưng giới hạn thời gian sử dụng.

II. CÁC VERSION CỦA SQL SERVER

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại chươngtrình nguồn gốc hoàn toàn mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên version 8.0 (hay SQL Server 2000) thì những cải tiến chủ yếu là mở rộng các tính năng về Web và làm cho SQL Server 2000 đáng tin cậy hơn.

Một điểm đặc biệt đáng lưu ý ở version 2000 chính là tính Multiple-Instance (đa thể hiện). Bạn có thể cài đặt (install) Version 2000 chung với các Version trước mà không cần phải gỡ (uninstall) chúng khỏi hệ thống. Nghĩa là bạn có thể chạy song song version 6.5 hoặc 7.0 với Version 2000 trên cùng một máy (điều này không thể xảy ra với các Version trước đây). Khi đó Version cũ trên máy bạn là một Instance ngầm định (Default Instance) còn Version 2000 mới cài đặt sẽ là Instance được đặt tên (Named Instance).

III. NHỮNG THÀNH PHẦN QUAN TRỌNG CỦA SQL SERVER 2000

SQL Server 2000 được cấu tạo bởi nhiều thành phần như *Relational Database Engine, Analysis Service và English Query,...* Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

3.1. Relational Database Engine - Bộ máy cơ sở dữ liệu quan hệ

Đây là một bộ máy có khả năng chứa data ở các quy mô khác nhau dưới dạng table và support tất cả các kiểu kết nối thông dụng của Microsoft như ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Ngoài ra nó còn có khả năng tự điều chỉnh (tune up), Ví dụ như sử dụng thêm các tài nguyên (resource) của máy khi cần và trả lại tài nguyên cho hệ điều hành khi một user log off.

3.2. Replication - Cơ chế tạo bản sao

Giả sử bạn có một database dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Bạn muốn có một database giống y hệt như thế trên một server khác để chạy báo cáo (report database) (cách làm này thường dùng để tránh ảnh hưởng đến hiệu suất làm việc của server chính). Vấn đề là report server đó cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. Chúng ta không thể dùng cơ chế *backup* and

II. CÁC VERSION CỦA SQL SERVER

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại chươngtrình nguồn gốc hoàn toàn mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên version 8.0 (hay SQL Server 2000) thì những cải tiến chủ yếu là mở rộng các tính năng về Web và làm cho SQL Server 2000 đáng tin cậy hơn.

Một điểm đặc biệt đáng lưu ý ở version 2000 chính là tính Multiple-Instance (đa thể hiện). Bạn có thể cài đặt (install) Version 2000 chung với các Version trước mà không cần phải gỡ (uninstall) chúng khỏi hệ thống. Nghĩa là bạn có thể chạy song song version 6.5 hoặc 7.0 với Version 2000 trên cùng một máy (điều này không thể xảy ra với các Version trước đây). Khi đó Version cũ trên máy bạn là một Instance ngầm định (Default Instance) còn Version 2000 mới cài đặt sẽ là Instance được đặt tên (Named Instance).

III. NHỮNG THÀNH PHẦN QUAN TRỌNG CỦA SQL SERVER 2000

SQL Server 2000 được cấu tạo bởi nhiều thành phần như *Relational Database Engine, Analysis Service và English Query,...* Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

3.1. Relational Database Engine - Bộ máy cơ sở dữ liệu quan hệ

Đây là một bộ máy có khả năng chứa data ở các quy mô khác nhau dưới dạng table và support tất cả các kiểu kết nối thông dụng của Microsoft như ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Ngoài ra nó còn có khả năng tự điều chỉnh (tune up), Ví dụ như sử dụng thêm các tài nguyên (resource) của máy khi cần và trả lại tài nguyên cho hệ điều hành khi một user log off.

3.2. Replication - Cơ chế tạo bản sao

Giả sử bạn có một database dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Bạn muốn có một database giống y hệt như thế trên một server khác để chạy báo cáo (report database) (cách làm này thường dùng để tránh ảnh hưởng đến hiệu suất làm việc của server chính). Vấn đề là report server đó cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. Chúng ta không thể dùng cơ chế *backup* and

restore trong trường hợp này. Thế thì phải làm sao? Lúc đó cơ chế replication của SQL Server sẽ được sử dụng để bảo đảm cho dữ liệu ở 2 database được đồng bộ (synchronized).

3.3. Data Transformation Service (DTS) - Dịch vụ truyền dữ liệu

Giả sử trong một công ty lớn, dữ liệu được chứa trong nhiều nơi khác nhau và ở các dạng khác nhau. Cụ thể như chứa trong Oracle, DB2 (của IBM), SQL Server, Microsoft Access,...

Khi có nhu cầu di chuyển data giữa các server này (migrate hay transfer) hoặc muốn định dạng (format) nó trước khi lưu vào database khác, khi đó bạn sẽ thấy DTS giúp bạn giải quyết công việc trên một cách dễ dàng.

3.4. Analysis Service - Dịch vụ phân tích dữ liệu

Microsoft cũng cấp một công cụ rất mạnh giúp cho việc phân tích dữ liệu trở nên dễ dàng và hiệu quả bằng cách dùng khái niệm hình lập phương nhiều chiều (multi- dimension cubes) và kỹ thuật khai thác dữ liệu.

3.5. English Query - Truy vấn bằng tiếng Anh

Đây là một dịch vụ giúp cho việc truy vấn dữ liệu bằng tiếng Anh thuần.

3.6. Meta Data Service - Dịch vụ siêu dữ liệu

Dịch vụ này giúp cho việc lưu trữ và thao tác với siêu dữ liệu (meta data) dễ dàng hơn. Meta data là những thông tin mô tả về cấu trúc của dữ liệu trong database như data thuộc loại nào: String hay Integer..., một cột nào đó có phải là Primary key hay không?...

Bởi vì những thông tin này cũng được chứa trong database nên cũng là một dạng data, nhưng để phân biệt với data "chính thống", người ta gọi nó là Meta Data. Chúng ta có thể tìm hiểu kỹ hơn vấn đề này trong SQL Server Books Online.

3.7. SQL Server Books Online - Sách hướng dẫn về SQL

Đây là quyển sách hướng dẫn khá chi tiết về SQL Server.

3.8. SQL Server Tools - Các công cụ quản trị hệ thống

- *Enterprise Manager* : Đây là một công cụ cho ta thấy toàn cảnh hệ thống cơ sở dữ liệu một cách rất trực quan. Nó rất hữu ích đặc biệt cho người mới học và không thông thạo lắm về SQL.

- *Query Analyzer* : Đối với một người quản trị cơ sở dữ liệu (DBA) giỏi thì hầu như chỉ cần công cụ này là có thể quản lý cả một hệ thống database mà không cần đến những thứ khác. Đây là một môi trường làm việc khá tốt vì ta có thể đánh bất kỳ câu lệnh SQL nào và chạy ngay lập tức, đặc biệt là nó giúp cho ta debug các stored procedure – thủ tục lưu trữ (Stored Procedure là một nhóm câu lệnh Transact-SQL đã được biên dịch và chứa trong SQL Server dưới một tên nào đó và được xử lý như một đơn vị chứ không phải nhiều câu SQL riêng lẻ) dễ dàng.
- *SQL Profiler* : Nó có khả năng "chụp" (capture) lại tất cả các sự kiện hay hoạt động diễn ra trên một SQL server và lưu lại dưới dạng text file rất hữu dụng trong việc kiểm soát hoạt động của SQL Server.
- Ngoài một số công cụ trực quan như trên chúng ta còn có thể dùng đến các lệnh osql và bcp (bulk copy) trong Comand Prompt (dấu nhắc lệnh).

Chương 7

TÌM HIỂU VỀ NGÔN NGỮ TRANSACT-SQL

I. SƠ LƯỢC VỀ TRANSACT SQL (T-SQL)

Transact-SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute) được sử dụng trong SQL Server khác với P-SQL (Procedural- SQL) dùng trong Oracle.

1.1. Một số khái niệm cơ bản

- *Table* (bảng): là cấu trúc lưu trữ cơ bản nhất trong hệ quản trị CSDL quan hệ (RDBMS), nó bao gồm 1 hoặc nhiều cột và 0 hoặc nhiều hàng.
- *Row* (hàng): là tổ hợp những giá trị của cột trong bảng. Một hàng còn có thể được gọi là một record.
- *Column* (cột): hiển thị một loại dữ liệu trong bảng, Ví dụ tên phòng ban trong bảng phòng ban. Người ta thể hiện nó thông qua tên cột và giữ số liệu dưới các kiểu và kích cỡ nhất định.
- *Field* (trường): là giao của cột và dòng. Field chính là nơi chứa dữ liệu. Nếu không có dữ liệu trong trường người ta nói trường có giá trị là không.
- *Primary Key* (khóa chính): là một cột hoặc một tập các cột xác định tính duy nhất của các hàng ở trong bảng. Ví dụ mã phòng ban. khoá chính nhất thiết phải có số liệu.
- *Foreign Key* (khóa ngoại): là một cột hoặc một tập các cột tham chiếu tới chính bảng đó hoặc một bảng khác. khoá ngoại xác định mối quan hệ giữa các bảng.
- *Constraint* (ràng buộc): là các ràng buộc dữ liệu. Ví dụ: khoá ngoại, khoá chính...

1.2. Tổng quan về một số lệnh SQL thông dụng

Lệnh	Mô tả
SELECT	Là lệnh thông dụng nhất, dùng để lấy, xem dữ liệu trong CSDL.
INSERT UPDATE DELETE	Là 3 lệnh dùng để nhập thêm những hàng mới, thay đổi nội dung dữ liệu trên các hàng hay xoá các hàng trong table. Những lệnh này được gọi là các lệnh thao tác dữ liệu DML (Data Manipulation Language).
CREATE ALTER DROP TRUNCATE	Là 4 lệnh dùng để thiết lập, thay đổi hay xoá bỏ cấu trúc dữ liệu như là table, view, index. Những lệnh này được gọi là các lệnh định nghĩa dữ liệu DDL (Data Definition Language).
GRANT REVOKE DENY	3 lệnh này dùng để gán hoặc huỷ các quyền truy nhập vào CSDL Oracle và các cấu trúc bên trong nó. Những lệnh này được gọi là các lệnh điều khiển dữ liệu DCL (Data Control Language.)

II. CÚ PHÁP CỦA T-SQL

2.1. Identifiers (định danh, tên)

Đây chính là tên của các đối tượng cơ sở dữ liệu (database object). Nó dùng để xác định một đối tượng nào đó (table, view, stored procedure, index,...). Ví dụ: TableX, KeyCol, Description đều là những identifiers:

CREATE TABLE TableX

(KeyCol INT PRIMARY KEY, Description NVARCHAR(80))

Có hai loại định danh (Identifier). Một loại là định danh thông thường (*Regular Identifier*) và một loại gọi là từ định danh phân định (*Delimited Identifier*), loại này cần có dấu " " hay dấu [] để ngăn cách. Delimited Identifier được dùng đối với các chữ trùng với từ khóa của SQL Server (reserved keyword) hoặc các chữ có khoảng trống.

Ví dụ:

SELECT * FROM [My Table] WHERE [Order] = 10

Trong ví dụ trên, chữ Order trùng với keyword Order còn chữ My Table không viết liền nhau nên cần đặt trong dấu ngoặc vuông [].

2.2. Expressions (Biểu thức)

Các biểu thức trong T-SQL thường có dạng:

Identifier + Operators (như +,-,*,/,=...) + Value

2.3. Comments (Chú thích)

T-SQL dùng -- ... -- để đánh dấu phần chú thích cho câu lệnh đơn và dùng /*...*/ để chú thích cho một nhóm lệnh.

2.4. Lệnh USE

Dùng thay đổi ngữ cảnh CSDL để chỉ định CSDL mà các câu lệnh tiếp theo sẽ có tác dụng trên đó. Tức là khai báo rằng ta sẽ dùng CSDL nào cho các lệnh phía sau nó. Cú pháp :

USE {database}

Trong đó *database* là tên của CSDL.

2.5. Variables (Biến)

Biến trong T-SQL cũng có chức năng tương tự như trong các ngôn ngữ lập trình khác, nghĩa là cần khai báo loại dữ liệu trước khi sử dụng. Biến được bắt đầu bằng dấu @. Riêng đối với các biến toàn cục (global variable) thì dùng dấu @@.

Ví dụ:

USE Northwind

DECLARE @TBLNHANVIENID Var INT SET
@TBLNHANVIENIDVar = 3

SELECT * FROM NhanViens

WHERE NhanVienID = @TBLNHANVIENIDVar + 1

Chú ý: CSDL Northwind là CSDL mẫu có sẵn sau khi ta cài SQL Server 2000).

2.6. Các kiểu dữ liệu

2.6.1. Kiểu nguyên

bigint

Kiểu số nguyên từ -2^{63} (tức -9.223.372.036.854.775.808) đến $2^{63}-1$ (tức 9.223.372.036.854.775.807). Dùng 8 byte để lưu trữ.

int

Kiểu số nguyên từ -2^{31} (-2.147.483.648) đến $2^{31} - 1$ (2.147.483.647). Dùng 4 byte để lưu trữ.

smallint

Kiểu số nguyên từ 2^{15} (-32.768) đến 2^{15} - 1 (32.767). Dùng 2 byte để lưu trữ.

tinyint

Kiểu số nguyên từ 0 đến 255. Dùng 1 byte đểlưu trữ.

2.6.2. Kiểu thực dấu phẩy tĩnh

decimal

Là kiểu số thập phân từ -10^38 +1 đến 10^38 –1. Khai báo :

decimal[(*p*[, *s*])] : p là số chữ số thập phân ở cả 2 phía trái và phải của dấu chấm thập phân

s là số chữ số thập phân bên vế phải của dấu chấm thập phân.

Tuỳ vào *p*, dung lượng nhớ cần thiết cho kiểu dữ liệu này sẽ khác nhau :

Số chữ số thập phân (p)	Số byte lưu trữ
1 – 9	5
10-19	9
20-28	13
29-38	17

numeric

Hoàn toàn tương tự như kiểu decimal. Khai báo: numeric[(*p*[, *s*])].

2.6.3. Kiểu thực dấu phẩy động

float

Kiểu số thực dấu phẩy động từ -1.79E + 308 đến 1.79E + 308. Khai báo :

float [(*n*)] : n là số bit dùng để lưu phần định trị của số thực dấu phẩy động theo cách ký hiệu khoa học. n là số nguyên có giá trị từ 1 đến 53 thể hiện độ chính xác (hay số chữ số biểu diễn được của kiểu số thực dấu phẩy động) :

Float (53) tương đương với kiểu Double Precision.

n	Độ chính xác	Kích thước lưu trữ
1 - 24	7 chữ số	4 bytes
25 - 53	15 chữ số	8 bytes

real

Kiểu số thực dấu phẩy động từ -3.40E + 38 đến 3.40E + 38. Dùng 4 byte để lưu trữ. Real tương đương với Float (24).

2.6.4. Kiểu tiền tệ

money

Dữ liệu kiểu tiền tệ có giá trị từ -2^{63} (-922 337 203 685 477.5808) đến $2^{63} - 1$ (+922 337 203 685 477.5807). Dùng 8 byte để lưu trữ.

smallmoney

Dữ liệu kiểu tiền tệ có giá trị từ -214 748.3648 đến +214 748.3647. Dùng 4 byte để lưu trữ.

2.6.5. Kiểu ngày giờ

datetime

Kiểu dữ liệu thời gian từ 1/1/1753 đến 31/12/9999 với độ chính xác là 3% giây (3.33 mili giây).

Thời gian thực	Thời gian được làm tròn theo kiểu datetime
01/01/98 23:59:59.999	1998-01-02 00:00:00.000
01/01/98 23:59:59.995, 01/01/98 23:59:59.996, 01/01/98 23:59:59.997, hoặc 01/01/98 23:59:59.998	1998-01-01 23:59:59.997
01/01/98 23:59:59.992, 01/01/98 23:59:59.993, 01/01/98 23:59:59.994	1998-01-01 23:59:59.993
01/01/98 23:59:59.990, hoặc 01/01/98 23:59:59.991	1998-01-01 23:59:59.990

smalldatetime

Kiểu dữ liệu ngày và thời gian từ 1/1/1900 đến 6/6/2079 với độ chính xác là 1 phút. Dữ liệu kiểu smalldatetime có giá trị từ 29.998 giây trở xuống sẽ bị làm tròn xuống đến phút gần nhất. Ngược lại, dữ liệu kiểu smalldatetime có giá trị từ 29.999 giây trở lên sẽ được làm tròn lên đến phút gần nhất.

Ví dụ :

Lệnh :

SELECT CAST('2000-05-08 12:35:29.998' AS smalldatetime)

sẽ trả về kết quả 12:35

Lệnh :

SELECT CAST('2000-05-08 12:35:29.999' AS smalldatetime)

sẽ trả về kết quả 12:36

(hàm CAST là hàm chuyển đổi dạng dữ liệu).

2.6.6. Kiểu xâu ký tự

char

Kiểu xâu ký tự phi Unicode (non-Unicode) có độ dài cố định, với chiều dài lớn nhất là 8,000 kí tự. Khai báo :

char[(*n*)]: n là độ dài xâu và cũng chính là số byte nhớ cần để lưu trữ xâu (0<n<8001). Ngầm định n = 1 khi không khai báo trong định nghĩa dữ liệu hay khai báo biến. Nếu xâu nhập vào có chiều dài nhỏ hơn n thì sẽ tự động bị thêm vào phần ký tự trống đằng sau sao cho độ dài đúng bằng n.

varchar

Tương tự như kiểu char nhưng có độ dài thay đổi. Vì thế sẽ không lưu phần ký tự trống phía sau nếu xâu nhập vào có độ dài nhỏ hơn độ dài đã khai báo.

varchar[(*n*)] : 0<n<8001.

<u>text</u>

Dữ liệu kiểu văn bản với độ dài tối đa lên tới 2^{31} - 1 (2.147.483.647) ký tự.

<u>nchar</u>

Kiểu xâu ký tự Unicode độ dài cố định có độ dài tối đa là 4.000 ký tự. Khai báo :

nchar(*n*) : độ dài tối đa là n ký tự và dung lượng nhớ dùng để lưu trữ kiểu dữ liệu này là 2n byte.

<u>nvarchar</u>

Giống như nchar nhưng độ dài xâu không cố định. Độ dài tối đa cũng là 4000 ký tự.

ntext

Dữ liệu kiểu văn bản Unicode với độ dài tối đa là $2^{30} - 1$ (1.073.741.823) ký tự.

image

Dữ liệu kiểu hình ảnh.

2.7. Một số hàm của T-SQL

2.7.1 Các hàm về số

- ROUND(n,m) cho giá trị làm tròn của n đến chữ số thập phân thứ m sau dấu chấm thập phân.
- CEILING (n) cho số nguyên nhỏ nhất >= n.
- FLOOR(n) cho số nguyên lớn nhất <= n (phần nguyên của n).
- POWER(n,m) cho lũy thừa bậc m của n.
- EXP(n) cho giá trị của e^n.
- SQRT(n) cho căn bậc 2 của n, n>=0.
- SIGN(n) cho dấu của n.

  n<0 có SIGN(n)= -1

  n=0 có SIGN(n)= 0

  n>0 có SIGN(n)= 1
- ABS(n) cho giá trị tuyệt đối.
- LOG(n) cho logarit tự nhiên của n.
- SIN(n) cho sin của n (n tính bằng radian).
- COS(n) cho cosin của n (n tính bằng radian).
- TAN(n) tang của n (n tính bằng radian).

2.7.2. Các hàm về xâu

- LOWER(char) cho chuỗi ký tự viết thường (không viết hoa).
- LTRIM(char) bỏ các ký tự trống bên trái của xâu ký tự.
- REPLACE(char,search_string,replacement_string): Thay tất cả các chuỗi search_string có trong chuỗi char bằng chuỗi replacement_string.
- RTRIM(char) bỏ các ký tự trống bên phải.
- SUBSTRING(char, start [,length]): Cho chuỗi con của chuỗi char bắt đầu từ vị trí start có độ dài length, nếu không chỉ length thì lấy cho đến cuối chuỗi.
- UPPER(char) cho chuỗi chữ hoa của chuỗi char.

- LEN (char) cho chiều dài của chuỗi char.

2.7.3. Các hàm ngày

- DAY (*date*) trả về một số nguyên là ngày của *date*.
- MONTH (*date*) trả về một số nguyên là tháng của *date*.
- YEAR (*date*) trả về một số nguyên là năm của *date*.
- GETDATE ( ) trả về ngày giờ hệ thống của SQL Server.

Hàm chuyển đổi kiểu dữ liệu

CAST (*expression* AS *data_type*)

trả về giá trị của *expression* theo kiểu *data_type*.

CONVERT (*data_type* [(*length*)] , *expression*)

trả về giá trị của *expression* theo kiểu *data_type* có độ dài length.

III. NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU - DDL (DATA DEFINITION LANGUAGE)

Đây là những lệnh dùng để quản lý các thuộc tính của một database như định nghĩa các hàng hoặc cột của một table, hay vị trí data file của một database... Thường có những dạng sau :

- Create *Tên đối tượng*
- Alter *Tên đối tượng*
- Drop *Tên đối tượng*

Trong đó đối tượng có thể là một table, view, stored procedure, indexes...

3.1. Tạo cơ sở dữ liệu - CREATE DATABASE

Khi tạo một cơ sở dữ liệu, một không gian cơ sở dữ liệu cũng tự động được tạo ra và các bảng (table) được đặt trong không gian cơ sở dữ liệu này.

Cú pháp :

```
CREATE DATABASE database_name
ON
( [NAME = logical_file_name ,]
[FILTENNV = 'os_file_name']
[, SIZE = size]
[, MAXSIZE = {max_size | UNLIMITED}]
[, FILEGROWTH = growth_increment] )
```

LOG ON

([NAME = *logical_file_name* ,]

[FILTENNV = '*os_file_name*']

[, SIZE = *size*]

[, MAXSIZE = {*max_size* | UNLIMITED}]

[, FILEGROWTH = *growth_increment*])

Trong đó:

database_name: tên CSDL.

logical_file_name: là tên file logic mà các câu lệnh T-SQL khi được thực hiện sẽ tham chiếu đến.

Os_file_name : đường dẫn thực đến file CSDL.

size : kích thước ban đầu của file CSDL.

Max_size : kích thước tối đa của file CSDL.

Growth_incremen : mức độ tăng trưởng dung lượng file dữ liệu mỗi khi cần nới rộng file CSDL (ngầm định là MB).

Ví dụ: lệnh CREAT DATABASE sau sẽ tạo ra một CSDL tên là NhanSu:

CREATE DATABASE *NhanSu*

ON

(NAME = NhanSu_dat,

FILTENNV = 'c:\program files\microsoft sql server
\mssql\data\NhanSudat.mdf',

SIZE = 10,

MAXSIZE = 50,

FILEGROWTH = 5)

LOG ON

(NAME = NhanSu_log,

FILTENNV = 'c:\program files\microsoft sql server
\mssql\data\NhanSulog.ldf',

SIZE = 5MB,

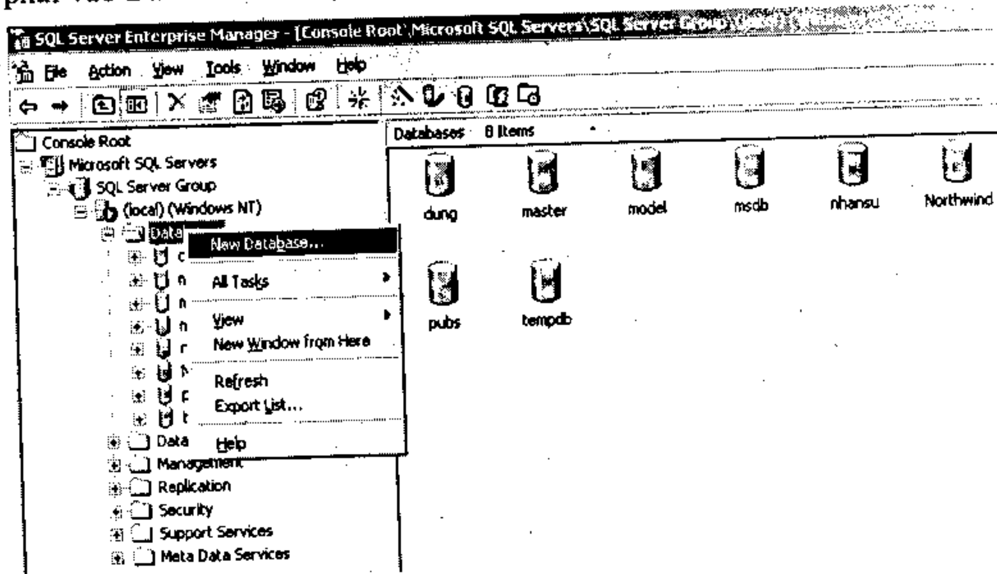
MAXSIZE = 25MB,

FILEGROWTH = 5MB)

Chú ý: Cũng có thể tạo CSDL bằng lệnh Create Database đơn giản sau đây, khi đó các thông số sẽ được chọn ngầm định :

CREAT DATABASE NhanSu

Hoặc có thể tạo CSDL bằng công cụ SQL Server Enterprise Manager của SQL Server 2000 như sau (tức là thông qua hệ thực đơn): nháy chuột phải vào Database và chọn New Database rồi đặt tên cho CSDL này.



3.2. Tạo bảng - CREATE TABLE

Cú pháp như sau:

CREATE TABLE table_name (col_name [Data_type] [col_constraint])

Trong đó:

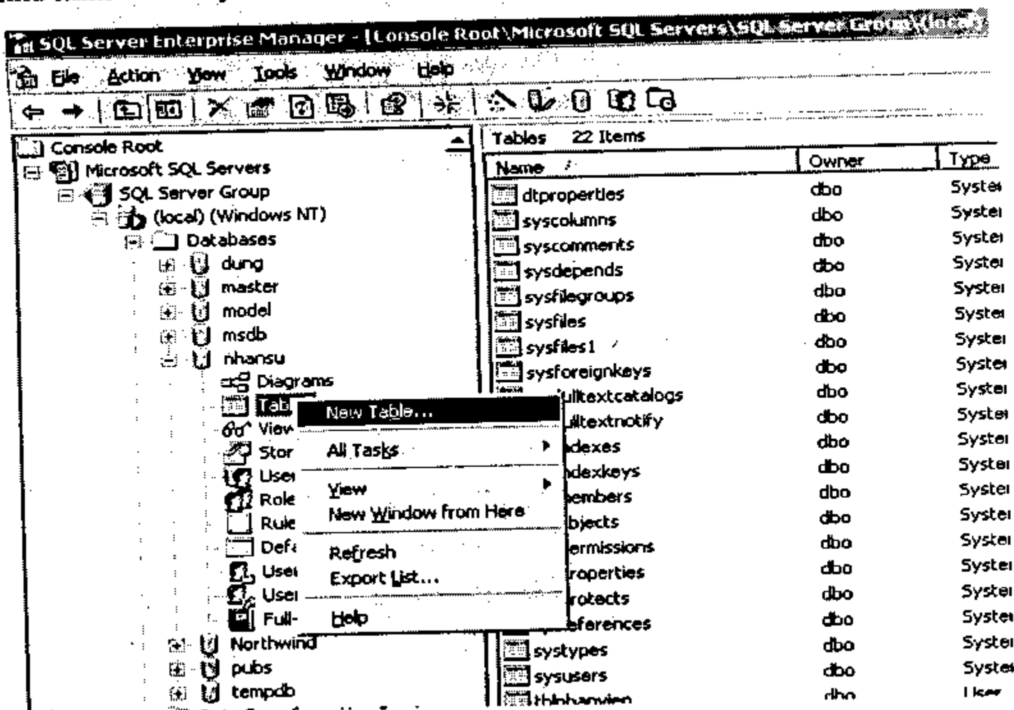
Table_name	: Tên table cần tạo
col_name	: Tên column trong table
data_type	: Kiểu dữ liệu của column
col_constraint	: Ràng buộc của bản thân column

Ví dụ :

Ví dụ lệnh Create Table sau sẽ tạo ra một table tên Mat_hang với 3 cột Mahang, TenCongTy, so_lg :

```
USE Nhansu
CREATE TABLE Mat_Hang
Mahang int NOT NULL,
TenCongTy varchar(40) NOT NULL,
so_lg int)
```

Cũng có thể tạo bảng bằng chính công cụ SQL Server Enterprise Manager của SQL Server 2000 bằng cách : truy cập vào CSDL (chẳng hạn như CSDL nhân sự), nhấn chuột phải vào mục Table rồi chọn New Table... như hình dưới đây :



3.3. Thay đổi định nghĩa bảng - ALTER TABLE

- Thêm cột :

ALTER TABLE table_name ADD Column_name Data_type

- Thay đổi định nghĩa cột :

```
ALTER TABLE table_name ALTER COLUMN Column_name
Data_type
```

- Huỷ bỏ cột :

```
ALTER TABLE table_name DROP COLUMN Column_name
```

Ví dụ 1: thêm cột ColAdd kiểu dữ liệu char(10) vào bảng nhân viên :

```
ALTER TABLE tblnhanvien ADD ColAdd CHAR(10)
```

Ví dụ 2: sửa ràng buộc char(50) của cột tên nhân viên thành char(55) :

```
ALTER TABLE tblnhanvien ALTER COLUMN tennv
CHAR(55)
```

Ví dụ 3: xoá cột ColAdd trong bảng nhân viên :

```
ALTER TABLE tblnhanvien DROP COLUMN ColAdd
```

3.4. Huỷ bảng - DROP TABLE

Lệnh DROP TABLE sẽ hoàn toàn xóa table khỏi database, nghĩa là cả định nghĩa của table và data bên trong table đều biến mất (khác với lệnh Delete chỉ xóa data nhưng table vẫn tồn tại).

Cú pháp :

```
DROP TABLE Table_name
```

Ví dụ :

```
USE Nhansu
DROP TABLE Tblnhanvien
```

3.5. Xoá hàng - TRUNCATE

Lệnh Truncate sẽ xóa tất cả các hàng của bảng mà không cần có mệnh đề WHERE (khác với lệnh Delete). Cú pháp :

```
TRUNCATE TABLE table_name
```

Ví dụ :

```
USE Nhansu
TRUNCATE TABLE tblnhanvien
```

IV. NGÔN NGỮ ĐIỀU KHIỂN DỮ LIỆU - DCL (DATA CONTROL LANGUAGE)

Đây là những lệnh quản lý các quyền truy cập lên từng đối tượng (table, view, stored procedure...).

4.1. Lệnh Grant

Là lệnh cho phép user nào đó có quyền thực hiện lệnh trên một bảng nào đó của CSDL đối với các user. Cú pháp :

```
USE Data_Base_Name
GRANT {ALL | statement_list} ON Table_name
TO security_account_list
```

trong đó : ALL : là từ khoá chỉ rằng tất cả mọi lệnh đều bị cấm.

Statement_list : danh sách các lệnh bị cấm.

Security_account_list : danh sách các account bị cấm.

Ví dụ:

Lệnh sau sẽ cho phép user trong nhóm Public được quyền Select đối với table *Tblnhanvien* trong database NhanSu:

```
USE Nhansu
GRANT SELECT ON Tblnhanvien TO PUBLIC
```

4.2. Lệnh Deny

Trái ngược với Grant, lệnh Deny từ chối quyền thực hiện các lệnh trên một bảng nào của CSDL đối với các user.

Cú pháp:

```
USE Data_Base_Name
DENY {ALL | statement_list}  ON Table_name
TO security_account_list
```

Ví dụ:

Lệnh sau sẽ từ chối quyền Select đối với table Tblnhanvien trong database NhanSu của các user trong nhóm Public:

```
USE NhanSu
DENY SELECT ON Tblnhanvien TO PUBLIC
```

4.3. Lệnh Revoke

Lệnh Revoke sẽ vô hiệu hoá tác dụng của Grant và Deny trước đó

Cú pháp :

```
USE Data_Base_Name
REVOKE {ALL | statement_list} ON Table_name
TO security_account_list
```

Ví dụ:

```
USE NhanSu
REVOKE SELECT ON Tblnhanvien TO PUBLIC
```

V. NGÔN NGỮ THAO TÁC DỮ LIỆU- DML (DATA MANIPULATION LANGUAGE)

Đây là những lệnh phổ biến dùng để xử lý dữ liệu như Select, Update, Insert, Delete. Trong đó câu lệnh Select là lệnh phổ biến nhất và cũng là lệnh quan trọng nhất mà chúng ta cần bàn kỹ.

5.1. Lệnh truy vấn cơ bản - SELECT

```
SELECT [DISTINCT] {*,column [alias],...}
FROM table
```

Trong đó :

SELECT - trả lời câu hỏi lấy dữ liệu nào? (column, biểu thức...), trong mệnh đề SELECT cần có ít nhất 1 column.

FROM - trả lời câu hỏi lấy dữ liệu ở đâu? (table, view...)

DISTINCT - chỉ định hiển thị các dữ liệu không trùng nhau (lọc dữ liệu từ các hàng có cùng giá trị).

* - thay cho việc chỉ tên tất cả các column.

alias - đưa ra nhãn (tên giả của cột hiện trên cửa sổ kết quả View) của column hiển thị dữ liệu.

Ví dụ:

```
SELECT * FROM tblnhanvien
SELECT manv, tennv, maphong FROM tblnhanvien
```

Kết quả :

manv	tennv	maphong
1	Nguyen Tuan Dung	1
2	Tran Phuong Thao	1
3	Nguyen Tuan Manh	2
4	Nguyen Dinh Dao	2
5	ACBA_B	3
6	ACBAB	1
7	I am Null	NULL
8	Pham Hoang Lam	3

SELECT DISTINCT luong LuongNhanVien FROM tblnhanvien

Kết quả :

	LuongNhanVien
1	NULL
2	500.0
3	1000.0
4	1500.0
5	2000.0
6	3000.0
7	4000.0

5.2. Truy vấn dữ liệu có điều kiện

5.2.1. Mệnh đề ORDER BY

Cú pháp :

```
SELECT [DISTINCT] {*, column [alias],...}
FROM table;
[WHERE condition]
[ORDER BY expr/position [DESC/ASC]]
```

Mệnh đề ORDER BY dùng để sắp xếp dữ liệu hiển thị và phải đặt ở vị trí sau cùng của câu lệnh truy vấn.

Ví dụ:

```
SELECT TENNV, NGHENGHIEP, LUONG*12, MAPHONG
FROM TBLNHANVIEN
ORDER BY TENNV
```

Kết quả:

	TENNV	NGHENGHIEP	(No column name)	MAPHONG
1	ACBA_B	Cong nhan	12000.0	3
2	ACBAB	Cong nhan	36000.0	1
3	I am Null	NULL	NULL	NULL
4	Nguyen Dinh Dao	Ky su	24000.0	2
5	Nguyen Tuan Dung	Ky su	6000.0	1
6	Nguyen Tuan Manh	Cong nhan	18000.0	2
7	Pham Hoang Lam	Ky Su	48000.0	3
8	Tran Phuong Thao	Ky su	6000.0	1

Mệnh đề ORDER BY mặc định sắp xếp theo thứ tự tăng dần ASC[ENDING]:

Số bé trước

Ngày nhỏ trước

Ký tự theo bảng chữ cái

Để sắp xếp theo thứ tự ngược lại (giảm dần), đặt từ khoá DESC[ENDING] sau column cần sắp thứ tự. Ví dụ:

```
SELECT TENNV, NGHENGHIEP
FROM TBLNHANVIEN
ORDER BY TENNV DESC
```

Order nhiều column

Mệnh đề Order còn có thể sắp xếp nhiều column. Các column cần sắp xếp được viết thứ tự sau mệnh đề ORDER BY và cách bởi dấu phẩy (,). Column nào gần mệnh đề ORDER BY hơn có mức độ ưu tiên khi sắp xếp cao hơn. Chỉ định cách thức sắp xếp ASC/DESC được viết sau column cách bởi một dấu cách.

Ví dụ:

```
SELECT MAPHONG, NGHENGHIEP, TENNV, LUONG
FROM TBLNHANVIEN
ORDER BY MAPHONG, LUONG DESC
```

Kết quả:

	MAPHONG	NGHENGHIEP	TENNV	LUONG
1	NULL	NULL	I am Null	NULL
2	1	Cong nhan	ACBAB	3000.0
3	1	Ky su	Nguyen Tuan Dung	500.0
4	1	Ky su	Tran Phuong Thao	500.0
5	2	Ky su	Nguyen Dinh Dao	2000.0
6	2	Cong nhan	Nguyen Tuan Manh	1500.0
7	3	Ky Su	Pham Hoang Lam	4000.0
8	3	Cong nhan	ACBA_B	1000.0

Chú ý:

Có thể chỉ định sắp xếp theo thứ tự các column trong mệnh đề SELECT.

Ví dụ: sắp xếp ưu tiên theo thứ tự nghề nghiệp.

```
SELECT MAPHONG, NGHENGHIEP, TENNV, LUONG
FROM TBLNHANVIEN
ORDER BY 2
```

Kết quả:

	MAPHONG	NGHENGHIEP	TENNV	LUONG
1	NULL	NULL	I am Null	NULL
2	3	Cong nhan	ACBA_B	1000.0
3	1	Cong nhan	ACBAB	3000.0
4	2	Cong nhan	Nguyen Tuan Manh	1500.0
5	2	Ky su	Nguyen Dinh Dao	2000.0
6	1	Ky su	Nguyen Tuan Dung	500.0
7	1	Ky su	Tran Phuong Thao	500.0
8	3	Ky Su	Pham Hoang Lam	4000.0

5.2.2. Mệnh đề WHERE

Cú pháp :

```
SELECT [DISTINCT] {*, column [alias],...}
FROM table_name
[WHERE condition]
[ORDER BY expr/position {DESC/ASC}]
```

Mệnh đề WHERE dùng để đặt điều kiện cho toàn bộ câu lệnh truy vấn. Trong mệnh đề WHERE có thể có các thành phần:

- Tên column.
- Toán tử so sánh.
- Tên column, hằng số hoặc danh sách các giá trị.

Ví dụ:

```
SELECT MAPHONG, NGHENGHIEP, TENNV, LUONG
FROM TBLNHANVIEN
WHERE LUONG BETWEEN 1000 AND 2000
```

a) Truy vấn dữ liệu với nhiều điều kiện

Mệnh đề WHERE cho phép ghép được nhiều điều kiện thông qua các toán tử logic AND/OR. Toán tử AND yêu cầu dữ liệu phải thoả mãn cả hai điều kiện. Toán tử OR cho phép dữ liệu thoả mãn một trong hai điều kiện.

Ví dụ:

```
SELECT MAPHONG, NGHENGHIEP, TENNV, LUONG
FROM TBLNHANVIEN
WHERE LUONG BETWEEN 1000 AND 2000 AND NGHENGHIEP =
'ky su'
```

```
SELECT MAPHONG, NGHENGHIEP, TENNV, LUONG
FROM TBLNHANVIEN
WHERE LUONG BETWEEN 1000 AND 2000 OR NGHENGHIEP =
'Ky su'
```

```
SELECT MAPHONG, NGHENGHIEP, MANV, TENNV, LUONG
FROM TBLNHANVIEN
WHERE LUONG > 1500
AND NGHENGHIEP = 'cong nhan'OR NGHENGHIEP ='ky su'
```

b) Các toán tử

Toán tử so sánh:

= : Toán tử bằng hay tương đương.

!=, <> : Toán tử khác hay không tương đương.

!<, !> : Toán tử không bé hơn/lớn hơn.

> : Toán tử lớn hơn.

< : Toán tử nhỏ hơn.

>= : Toán tử lớn hơn hoặc bằng.

<= : Toán tử nhỏ hơn hoặc bằng.

Các toán tử logic:

NOT : Toán tử Phủ định mệnh đề.

AND : Toán tử Và, yêu cầu dữ liệu phải thoả mãn cả hai điều kiện.

OR : Toán tử Hoặc cho phép dữ liệu thoả mãn một trong hai điều kiện.

Các toán tử của SQL:

[NOT] BETWEEN x AND y : [Không] lớn hơn hoặc bằng x và nhỏ hơn hoặc bằng y.

IN (danh sách): thuộc bất kỳ giá trị nào trong danh sách.

x [NOT] LIKE y: Đúng nếu x [không] giống khung mẫu y. Các ký tự dùng trong khuôn mẫu:

Dấu gạch dưới (_)	: Chỉ một ký tự bất kỳ.
Dấu phần trăm (%)	: Chỉ một nhóm ký tự bất kỳ.
IS [NOT] NULL	: Kiểm tra giá trị [không] rỗng.
EXISTS	: Trả về TRUE nếu có tồn tại.

Ví dụ :

Chọn nhân viên có lương nằm trong khoảng 2000 và 3000:

```
SELECT * FROM tblnhanvien WHERE luong BETEEN 2000
AND 3000
```

Chọn nhân viên có lương bằng một trong hai giá trị 1400 hoặc 3000:

```
SELECT * FROM tblnhanvien WHERE luong IN (1400,3000)
```

Tìm tên phòng ban nếu phòng đó có nhân viên làm việc:

```
SELECT tenphong FROM tblphong
WHERE EXISTS (SELECT * FROM tblnhanvien
WHERE tblphong.maphong = tblnhanvien.maphong)
```

Tìm nhân viên có tên "Nguyen Van Hung":

```
SELECT * FROM tblnhanvien WHERE tennv LIKE 'Nguyen
Van Hung'
```

Để chọn những nhân viên có tên bắt đầu bằng "Nguyen":

```
SELECT * FROM tblnhanvien WHERE tennv LIKE
'nguyen%'
```

Để tìm những nhân viên mà tên có chuỗi "A_B":

```
SELECT tennv FROM tblnhanvien WHERE tennv LIKE
'%A\_B%' ESCAPE '\'
```

Vì ký hiệu "_" ngầm định dùng để đại diện cho một ký tự bất kỳ nên nếu không có mệnh đề ESCAPE, câu lệnh trên sẽ tìm tất cả các nhân viên tên AAB, ABB, ACB, v.v...

Nếu muốn ký hiệu "_" mang ý nghĩa nguyên thủy, tức là không còn đại diện cho ký tự bất kỳ nữa mà chỉ là một ký tự "_", ta đặt dấu "\" trước ký hiệu. Đồng thời khai báo thêm mệnh đề ESCAPE "\".

Ta cũng có thể dùng một ký tự bất kỳ thay cho "\". Chẳng hạn mệnh đề sau có cùng kết quả với mệnh đề trên:

```
SELECT tennv FROM tblnhanvien WHERE tennv LIKE
'%A^_B%' ESCAPE '^'
```

Ta gọi các ký tự như "\" hay "^" nói trên là các ký tự ESCAPE.

Chọn tất cả các nhân viên có mã phòng chưa được xác định :

```
SELECT * FROM tblnhanvien WHERE maphong IS NULL
```

5.2.3. Mệnh đề GROUP BY và HAVING

Cú pháp:

```
SELECT column_list
FROM table
[WHERE condition]
[GROUP BY column_list]
[HAVING condition]
[ORDER BY expr/position [DESC/ASC]]
```

Mệnh đề GROUP BY sẽ nhóm các dòng dữ liệu có cùng giá trị của cột theo thứ tự ưu tiên lần lượt. Ví dụ: *GROUP BY luong* nghĩa là sẽ nhóm các bản ghi select được mà có lương giống nhau vào thành một nhóm. Chú ý : column_list ở mệnh đề Select và mệnh đề Group By phải giống nhau (thứ tự các tên cột có thể khác nhau).

Ví dụ:

Lệnh sau sẽ cho kết quả trong cửa sổ View được sắp xếp theo thứ tự ưu tiên giới tính rồi đến mã phòng, cuối cùng mới đến tên nhân viên :

```
SELECT tennv,maphong,gioitinh FROM tblnhanvien
GROUP BY gioitinh,maphong,tennv
```

	tennv	maphong	gioitinh
1	I am Null	NULL	NULL
2	Nguyen Tuan Dung	1	nam
3	Nguyen Dinh Dao	2	nam
4	Nguyen Tuan Manh	2	nam
5	ACBA_B	3	nam
6	Pham Hoang Lam	3	nam
7	ACBAB	1	nu
8	Tran Phuong Thao	1	nu

Mệnh đề HAVING là mệnh đề đặt điều kiện lên các nhóm dữ liệu. Mệnh đề này khác mệnh đề WHERE ở chỗ mệnh đề WHERE đặt điều kiện cho toàn bộ câu lệnh SELECT. Còn HAVING đặt điều kiện lên các nhóm phần tử đã được GROUP BY.

Ví dụ :

Lệnh sau sẽ đưa ra các nhân viên thuộc phòng có mã bằng 1 theo thứ tự giới tính nam trước, nữ sau :

```
SELECT tennv,maphong,gioitinh FROM tblnhanvien
GROUP BY gioitinh,maphong,tennv
HAVING maphong = 1
```

	tennv	maphong	gioitinh
1	Nguyen Tuan Dung	1	nam
2	ACBAB	1	nu
3	Tran Phuong Thao	1	nu

Một số hàm tác động trên nhóm

Các hàm tác động trên nhóm các dòng dữ liệu tác động lên một tập hợp các các dòng dữ liệu. Gồm các hàm:

AVG([DISTINCT/ALL] *column_name*)	Giá trị trung bình của cột mang tên *column_name*, không kể giá trị NULL.
COUNT([DISTINCT/ALL] expr)	Số row có expr khác null
MAX([DISTINCT/ALL] *column_name*)	Giá trị lớn nhất của cột mang tên *column_name*.
MIN([DISTINCT/ALL] *column_name*)	Giá trị nhỏ nhất của cột mang tên *column_name*.
SUM([DISTINCT/ALL] *column_name*)	Tổng của của cột mang tên *column_name* không kể giá trị NULL

Từ khoá DISTINCT để chỉ ra rằng hàm chỉ tác động lên những giá trị khác nhau của cột (các giá trị bằng nhau chỉ được lấy một giá trị làm đại diện tham gia vào việc tính hàm).

Có hai cách để dùng các hàm này:

- Tác động trên toàn bộ các dòng dữ liệu của câu lệnh truy vấn.

- Tác động trên một nhóm dữ liệu cùng tính chất của câu lệnh truy vấn. Cùng tính chất được chỉ bởi mệnh đề :

```
[GROUP BY column_name]
[HAVING condition]
```

Ví dụ về tác động trên toàn bộ các dòng dữ liệu của câu lệnh truy vấn:

```
SELECT AVG(luong)
FROM tblnhanvien
```

/Tính mức lương trung bình của toàn bộ nhân viên /

```
SELECT MIN(luong)
FROM tblnhanvien
WHERE gioitinh = 'nam'
```

/Tính mức lương thấp nhất của nhân viên nam /

Ví dụ về tác động trên một nhóm dữ liệu cùng tính chất của câu lệnh truy vấn:

```
SELECT gioitinh, AVG(luong) LuongTB
FROM tblnhanvien
GROUP BY gioitinh
```

/ Tính mức lương trung bình của từng loại giới tính/

	gioitinh	LuongTB
1	NULL	NULL
2	nam	1100.0
3	nu	1750.0

5.2.4. Hiển thị nội dung dữ liệu từ nhiều bảng

Để liên kết trong mệnh đề WHERE phải chỉ rõ tên của các column và mệnh đề được đặt tương đương.

Ví dụ: `tblnhanvien.maphong = tblphong.maphong`

Các column trùng tên phải được chỉ rõ column đó nằm ở bảng nào thông qua tên hoặc qua alias. Tên trùng này có thể đặt trong các mệnh đề khác như SELECT, ORDER BY...

Ví dụ:

```
SELECT TBLPHONG.MAPHONG, TENNV, TENPHONG
```

```
FROM TBLNHANVIEN, TBLPHONG
WHERE TBLNHANVIEN.MAPHONG = TBLPHONG.MAPHONG
ORDER BY TBLPHONG.MAPHONG
```

Kết quả :

	MAPHONG	TENNV	TENPHONG
1	1	Nguyen Tuan Dung	Hanh Chinh
2	1	Tran Phuong Thao	Hanh Chinh
3	1	ACBAB	Hanh Chinh
4	2	Nguyen Tuan Manh	Ky Thuat
5	2	Nguyen Dinh Dao	Ky Thuat
6	3	ACBA_B	Van Thu
7	3	Pham Hoang Lam	Van Thu

```
SELECT A.MAPHONG, A.TENNV, B.TENPHONG
FROM TBLNHANVIEN A, TBLPHONG B
WHERE A.MAPHONG = B.MAPHONG
ORDER BY A.MAPHONG
```

Kết quả:

	MAPHONG	TENNV	TENPHONG
1	1	Nguyen Tuan Dung	Hanh Chinh
2	1	Tran Phuong Thao	Hanh Chinh
3	1	ACBAB	Hanh Chinh
4	2	Nguyen Tuan Manh	Ky Thuat
5	2	Nguyen Dinh Dao	Ky Thuat
6	3	ACBA_B	Van Thu
7	3	Pham Hoang Lam	Van Thu

Liên kết của bảng với chính nó:

Có thể liên kết bảng với chính nó bằng cách đặt alias.

Ví dụ:

```
SELECT e.TENNV TenNhanVien, e.LUONG
LuongCuaNhanVien,
        m.TENNV TenNguoiQuanLy, m.LUONG
LuongCuaNguoiQuanLy
FROM   TBLNHANVIEN e, TBLNHANVIEN m
```

```
WHERE  e.maql = m.manv AND e.LUONG < m.LUONG
```

Kết quả :

	TenNhanVien	LuongCuaNhanVien	TenNguoiQuanLy	LuongCuaNguoiQuanLy
1	Tran Phuong Thao	500.0	Nguyen Tuan Dung	5000.0
2	Nguyen Tuan Manh	1500.0	Nguyen Dinh Dao	2000.0
3	ACBA_B	1000.0	Pham Hoang Lam	4000.0
4	ACBAB	3000.0	Nguyen Tuan Dung	5000.0

5.2.5. Toán tử UNION

- UNION: Kết hợp kết quả của nhiều câu hỏi với nhau, chỉ giữ lại một đại diện cho các mẫu tin trùng nhau.

- UNION ALL: Kết hợp kết quả của nhiều câu hỏi với nhau, các mẫu tin trùng nhau cũng được lặp lại.

Ví dụ:

```
SELECT NGHENGHIEP FROM TBLNHANVIEN where maphong = 1
UNION
SELECT NGHENGHIEP FROM TBLNHANVIEN where maphong = 3
```

Kết quả :

	NGHENGHIEP
1	Cong nhan
2	Ky su

```
SELECT NGHENGHIEP FROM TBLNHANVIEN where maphong = 1
UNION ALL
SELECT NGHENGHIEP FROM TBLNHANVIEN where maphong = 3
```

Kết quả :

	NGHENGHIEP
1	Ky su
2	Ky su
3	Cong nhan
4	Cong nhan
5	Ky Su

Khi dùng Union phải chú ý hai vấn đề: số cột select ở 2 truy vấn phải bằng nhau và kiểu dữ liệu của các cột tương ứng phải tương thích với nhau.

5.2.6. Các lệnh truy vấn lồng nhau

Trong mệnh đề WHERE

Ví dụ: tìm những nhân viên làm cùng nghề với Tran Phuong Thao:

```
SELECT tennv,nghenghiep FROM tblnhanvien
WHERE nghenghiep = (SELECT nghenghiep FROM
 tblnhanvien WHERE TENNV = 'Tran Phuong Thao')
```

Kết quả :

	tennv	nghenghiep
1	Nguyen Tuan Dung	Ky su
2	Tran Phuong Thao	Ky su
3	Nguyen Dinh Dao	Ky su
4	Pham Hoang Lam	Ky Su

Trong mệnh đề HAVING

Ví dụ: tìm những phòng có mức lương trung bình lớn hơn mức lương trung bình của phòng 3:

```
   SELECT maphong, AVG(luong) LuongTrungBinh
 FROM tblnhanvien
   GROUP BY maphong
   HAVING AVG(luong) > (SELECT AVG(luong) FROM
 tblnhanvien WHERE maphong =3)
```

Kết quả :

	maphong	LuongTrungBinh
1	1	2833.3333333333335

5.2.7. Cấu trúc tổng quát của lệnh SELECT

```
   SELECT select_list
 [INTO new_table]
 FROM table_source
 [WHERE search_condition]
 [GROUP BY group_by_expression]
 [HAVING search_condition]
 [ORDER BY order_expression [ASC | DESC]]
```

5.3. Các lệnh thao tác dữ liệu khác

5.3.1. Chèn dữ liệu - INSERT

Để chèn một hàng mới vào bảng, ta dùng lệnh INSERT. Cú pháp như sau:

```
INSERT INTO table_name ([column, column, ...])
VALUES (value, value ...)
```

Ví dụ:

```
INSERT INTO tblphong (maphong, tenphong,
diadiem)
VALUES (5, 'MARKETING', 'tang3')
```

Chép dữ liệu từ table khác:

```
INSERT INTO table_name [(column, column...)]
SELECT select_list
FROM table_name
```

Ví dụ:

```
INSERT INTO TBLNHANVIEN_New (TENNV, LUONG)
SELECT TENNV, LUONG FROM TBLNHANVIEN WHERE LUONG >
1000
```

(Chú ý : bảng tblnhanvien_new phải được tạo rồi)

5.3.2. Chỉnh sửa dữ liệu - UPDATE

Để chỉnh sửa dữ liệu dùng lệnh UPDATE. Cú pháp như sau :

```
UPDATE table_name [alias]
SET column [,column...] = [expr, subquery]
[WHERE condition]
```

Trong đó :

expr, subquery là một biểu thức hay một câu truy vấn lồng.

Ví dụ 1: sửa nghề nghiệp và lương của nhân viên Tran Phuong Thao:

```
UPDATE TBLNHANVIEN
SET NGHENGHIEP = 'ThacSy', LUONG = LUONG * 2
WHERE TENNV = 'Tran Phuong Thao'
```

Ví dụ 2: sửa lương của nhân viên Tran Phuong Thao thành mức lương thấp nhất:

```
UPDATE TBLNHANVIEN
SET luong = (SELECT min(luong) FROM tblnhanvien)
WHERE TENNV = 'Tran Phuong Thao'
```

5.3.3. Xóa hàng dữ liệu – DELETE

Để xóa hàng trong bảng, ta dùng lệnh DELETE. Cú pháp như sau:

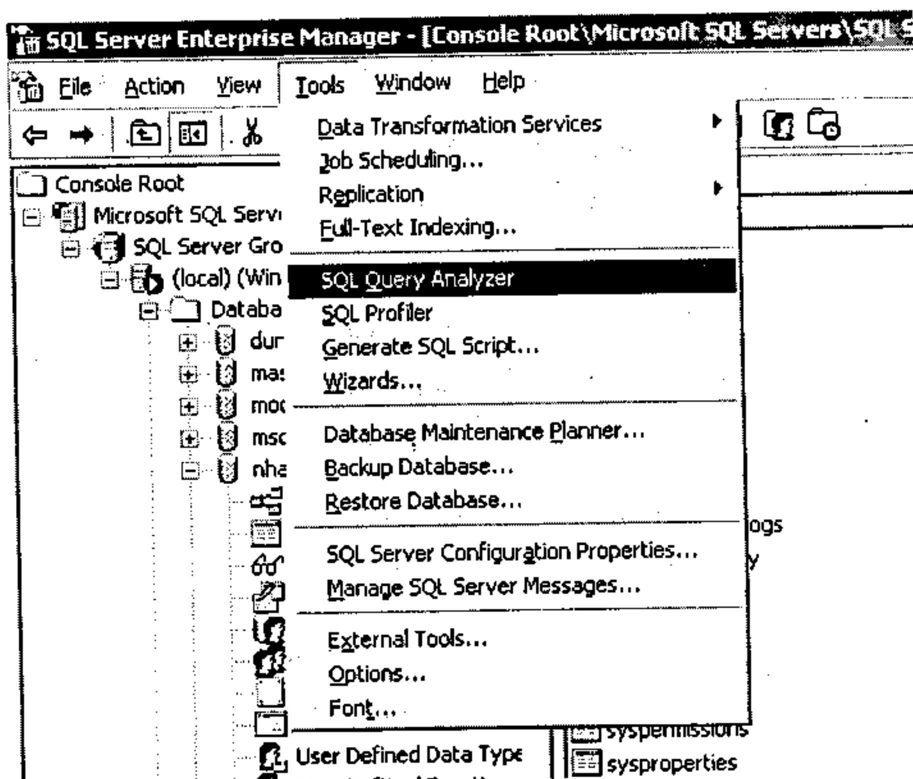
```
DELETE FROM table_name [WHERE condition]
```

Ví dụ: xoá tất cả các thông tin về nhân viên trong phòng 1:

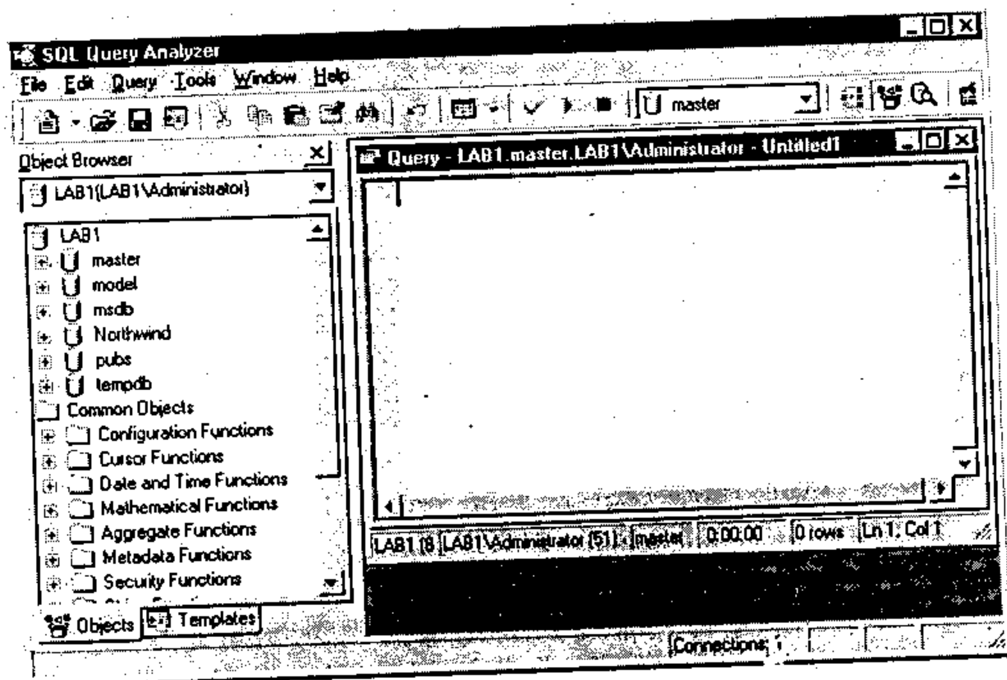
```
DELETE FROM tblnhanvien
WHERE maphong = 1
```

VI. THỰC THI CÁC CÂU LỆNH SQL

Để thực thi các câu lệnh SQL, chúng ta có thể dùng công cụ SQL Query Analyzer bằng cách: trên menu của SQL Enterprise Manager, chọn *Tools\SQL Query Analyzer* như hình dưới đây :



Sau đó cửa sổ SQL Query Analyzer sẽ xuất hiện :



6.1. Thực thi một câu lệnh đơn

Một câu lệnh SQL được phân ra thành các thành phần cú pháp như trên bởi một parser, sau đó SQL Optimizer (một bộ phận quan trọng của SQL Server) sẽ phân tích và tìm cách thực thi (Execute Plan) tối ưu nhất. Ví dụ như cách nào nhanh và tốn ít tài nguyên của máy nhất... và sau đó SQL Server Engine sẽ thực thi và trả về kết quả.

6.2. Thực thi một nhóm lệnh (Batches)

Khi thực thi một nhóm lệnh, SQL Server sẽ phân tích và tìm biện pháp tối ưu cho các câu lệnh như một câu lệnh đơn và ghi kế hoạch thực thi (execution plan) đã được biên dịch (compiled) vào trong bộ nhớ. Sau đó, nếu nhóm lệnh trên được gọi lại lần nữa thì SQL Server không cần biên dịch mà có thể thực thi ngay, điều này giúp cho một nhóm lệnh chạy nhanh hơn.

6.3. Lệnh GO

Lệnh này chỉ dùng để gửi một tín hiệu cho SQL Server biết đã kết thúc việc thực hiện một nhóm lệnh và yêu cầu thực thi. Nó vốn không phải là

một lệnh trong T-SQL mà là lệnh của các tiện ích osql, isql và của SQL Query Analyzer.

Xét ví dụ sau :

```
USE pubs
DECLARE @NmbrAuthors int
SELECT @NmbrAuthors = COUNT(*)
FROM authors
PRINT 'The number of authors as of ' +
 CAST(GETDATE() AS char(20)) + ' is ' +
 CAST(@NmbrAuthors AS char (10))
GO
PRINT CAST (@NmbrAuthors AS char (10))
```

Khi thực thi nhóm lệnh trên bằng công cụ SQL Query Analyzer, ở cửa sổ kết quả bên dưới, bạn sẽ nhận được thông báo yêu cầu khai báo biến *@NmbrAuthors* vì lệnh GO đã kết thúc việc thực hiện nhóm lệnh này và biến *@NmbrAuthors* đã được giải phóng:

The number of authors as of Jan 29 2004 2:06PM is 23

Server: Msg 137, Level 15, State 2, Line 1

Must declare the variable '@NmbrAuthors'.

Trong khi đó, nếu không có dòng lệnh GO, kết quả nhận được sẽ là:

The number of authors as of Jan 29 2004 2:06PM is 23

23

Như vậy, trong chương này, chúng ta đã tìm hiểu sơ lược về ngôn ngữ T-SQL, các thành phần tạo nên cú pháp của nó cũng như cách thực thi các lệnh của T-SQL.

Chương 8

CẤU TRÚC VÀ HOẠT ĐỘNG CỦA SQL SERVER 2000

I. CẤU TRÚC CỦA SQL SERVER 2000

Như đã trình bày ở trên, một trong những đặc điểm của SQL Server 2000 là Multiple-Instance, nên khi nói đến một (SQL) Server nào đó là ta nói đến một Instance của SQL Server 2000, thông thường đó là Default Instance. Một Instance của SQL Server 2000 có 4 system databases (cơ sở dữ liệu hệ thống) và một hay nhiều user database (cơ sở dữ liệu người dùng). Các system databases bao gồm:

- *Master* : Chứa tất cả những thông tin cấp hệ thống (system-level information) bao gồm thông tin về các database khác trong hệ thống như vị trí của các data files, các login account và các thiết đặt cấu hình hệ thống của SQL Server (system configuration settings).

- *TTBLNHANVIENdb* : Chứa tất cả những table hay stored procedure được tạm thời tạo ra trong quá trình làm việc bởi user hay do bản thân SQL Server engine. Các table hay stored procedure này sẽ biến mất khi khởi động lại SQL Server hay khi ta disconnect.

- *Model* : Database này đóng vai trò như một database mẫu (TTBLNHANVIENlate) cho các database khác. Nghĩa là khi một user database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures...) từ model database sang database mới vừa tạo.

- *Msdb* : Database này được SQL Server Agent sử dụng để hoạch định các báo động và các công việc cần làm (schedule alerts and NGHENGHIEPs).

1.1. Cấu trúc vật lý của một CSDL

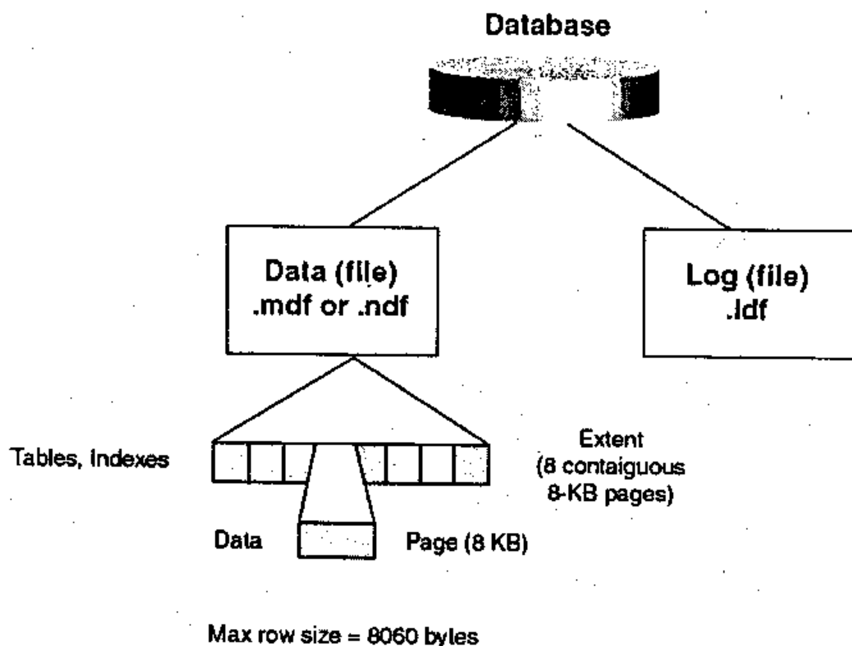
Mỗi một database trong SQL Server đều chứa ít nhất một data file chính (primary), có thể có thêm một hay nhiều data file phụ (Secondary) và một transaction log file:

- *Primary data file* (thường có phần mở rộng .mdf) : đây là file chính chứa data và những system tables.

- *Secondary data file* (thường có phần mở rộng .ndf) : đây là file phụ thường chỉ sử dụng khi database được phân chia để chứa trên nhiều đĩa.

- *Transaction log file* (thường có phần mở rộng .ldf) : đây là file ghi lại tất cả những thay đổi diễn ra trong một database và chứa đầy đủ thông tin để có thể roll back hay roll forward khi cần.

Data trong SQL Server được chứa thành từng Page 8KB và 8 page liên tục tạo thành một Extent như hình vẽ dưới đây:



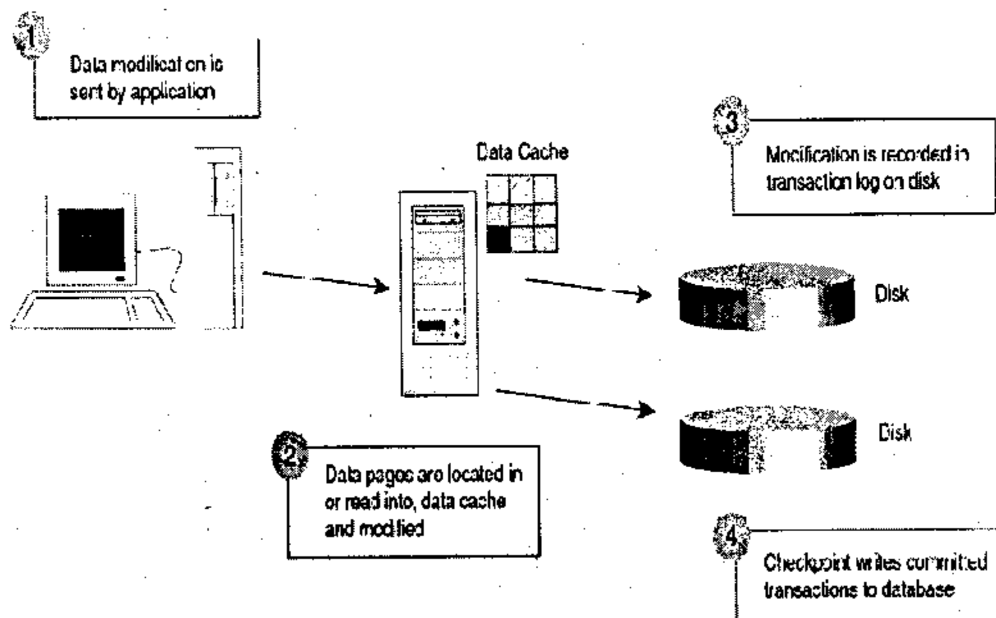
Trước khi SQL Server muốn lưu data vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các extents.

Có 2 loại Extents: Mixed Extents (loại hỗn hợp) dùng để chứa data của nhiều tables trong cùng một Extent và Uniform Extent (loại thuần nhất) dùng để chứa data của một table. Đầu tiên SQL Server dành các Page trong Mixed Extent để chứa data cho một table. Sau đó, khi data tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

1.2. Nguyên tắc hoạt động của Transaction Log

Transaction log file trong SQL Server dùng để ghi lại các thay đổi xảy ra trong database. Quá trình này diễn ra như sau: đầu tiên khi có một sự thay đổi data như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL

Server sẽ nạp (load) *data page* tương ứng vào memory (vùng bộ nhớ này gọi là *data cache*), sau đó dữ liệu trong data cache được thay đổi (những trang được thay đổi còn gọi là *dirty-page*). Tiếp theo mọi sự thay đổi đều được ghi vào transaction log file, quá trình này người ta gọi là *write-ahead log*. Cuối cùng thì một quá trình gọi là *Check Point Process* sẽ kiểm tra và ghi lại vào đĩa cứng tất cả những giao dịch (transaction) trên khi chúng đã được xác nhận (commited).



Ngoài Check Point Process những dirty-page còn được đưa vào đĩa bởi một *Lazy writer*. Đây là một chương trình chỉ quét qua phần data cache theo một chu kỳ nhất định sau đó lại ngừng hoạt động và chờ lần quét tới.

Ở đây có thể giải thích thêm một chút về khái niệm *transaction* trong database. Một transaction hay một giao dịch là một loạt các hoạt động xảy ra được xem như một đơn vị công việc (unit of work) nghĩa là hoặc thành công toàn bộ hoặc không làm gì cả. Sau đây là một ví dụ cổ điển về transaction:

Chúng ta muốn chuyển một số tiền $500 từ account A sang account B như vậy công việc này cần làm các bước sau:

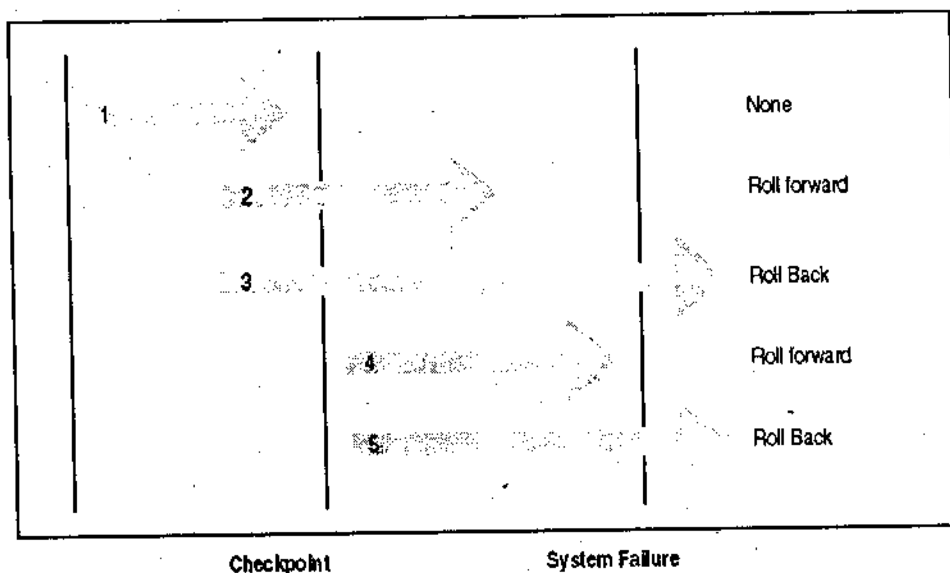
1. Trừ $500 ở account A

2. Cộng $500 vào account B

Tuy nhiên việc chuyển tiền trên phải được thực hiện dưới dạng một transaction, nghĩa là giao dịch chỉ được xem là hoàn tất (commited) khi cả hai bước trên đều thực hiện thành công. Nếu vì một lý do nào đó ta chỉ thực hiện được bước 1 (chẳng hạn như vừa xong bước 1 thì điện cúp hay máy bị treo) thì xem như giao dịch không hoàn tất và cần phải được phục hồi lại trạng thái ban đầu (roll back).

Thế thì *Check Point Process* hoạt động như thế nào để có thể đảm bảo cho việc thực thi một transaction mà không làm hỏng database?

Transaction Recovery



Trong hình vẽ trên, một transaction được biểu diễn bằng một mũi tên. Trục nằm ngang là trục thời gian. Giả sử một Check Point được đánh dấu vào thời điểm giữa transaction 2 và 3 như hình vẽ và sau đó sự cố xảy ra trước khi gặp một Check point kế tiếp. Như vậy khi SQL Server được restart nó sẽ dựa trên những thông tin ghi trong transaction log file để phục hồi data (xem hình vẽ).

Điều đó có nghĩa là SQL Server sẽ không cần làm gì cả đối với transaction 1 vì tại thời điểm Check point data đã được lưu vào đĩa rồi. Trong khi đó, transaction 2 và 4 sẽ được roll forward vì tuy đã được commited nhưng do sự cố xảy ra trước thời điểm check point kế tiếp nên data chưa kịp lưu vào đĩa. Tức là dựa trên những thông tin được ghi trên log file SQL Server hoàn toàn có đầy đủ cơ sở để viết vào đĩa cứng. Còn

transaction 3 và 5 thì chưa được commited (do bị down bất ngờ) cho nên SQL Server sẽ roll back hai transaction này dựa trên những gì được ghi trên log file.

1.3. Cấu trúc logic của một CSDL

Hầu như mọi thứ trong SQL Server được tổ chức thành những đối tượng (object). Ví dụ như tables, views, stored procedures, indexes, constraints,.... Những đối tượng hệ thống (system objects) trong SQL Server thường có bắt đầu bằng chữ sys hay sp. Trong phần này, chúng ta sẽ bàn sơ qua một số system object thông dụng trong SQL Server database:

System Stored Procedure	Ứng dụng
Sp_help ['object']	Cung cấp thông tin về một database object (table, view...) hay một data type.
Sp_helpdb ['database']	Cung cấp thông tin về một database cụ thể nào đó.
Sp_monitor	Cho biết độ bận rộn của SQL Server
Sp_spaceused *['object','updateusage']*	Cung cấp thông tin về các không gian đã được sử dụng cho một đối tượng nào đó.
Sp_who ['login']	Cho biết thông tin về một SQL Server user.

Ví dụ:

- Trong SQL Query Analyzer, gõ `sp_helpdb 'Northwind'` sẽ cho kết quả thông tin về cơ sở dữ liệu NorthWind có dạng như bảng dưới đây:

	name	db_size	owner	dbid	created	status	compatibility_level
1	Northwind	3.63 MB	sa	6	Aug 6 2000	Status=ONLINE, Updat...	80

	name	fileid	filename	filegroup	size	maxsize	growth	usage
1	Northwind	1	D:\Program Files\Mic...	PRIMARY	2688 KB	Unlimited	10%	data on
2	Northwind_log	2	D:\Program Files\Mic...	NULL	1024 KB	Unlimited	10%	log onl

- Hoặc sử dụng *sp_spaceused* như ví dụ sau:

```
USE Northwind
Go
sp_spaceused 'Customers'
```

sẽ cho biết thông tin về table Customer:

	name	rows	reserved	data	index_size	unused
1	Customers	91	104 KB	24 KB	80 KB	0 KB

II. NHỮNG ĐIỂM CẦN LƯU Ý KHI THIẾT KẾ CSDL

2.1. Tạo CSDL

Khi tạo ra một database, chúng ta phải lưu ý một số điểm sau: Đối với các hệ thống nhỏ mà ở đó vấn đề tốc độ của server không cần quan tâm thì chúng ta thường chọn các giá trị mặc định (default) cho Initial size, Automatically growth file. Nhưng trên một số server của các hệ thống lớn kích thước của database phải được người quản trị sơ sở dữ liệu (DBA) ước lượng trước tùy theo tầm cỡ của công việc, và thông thường người ta không chọn Autogrowth (tự động tăng trưởng) và Autoshrink (tự động nén). Vì sao ta không để SQL Server chọn một giá trị khởi đầu cho datafile và sau đó khi cần thì nó sẽ tự động nới rộng ra mà lại phải ước lượng trước? Nguyên nhân là nếu chọn Autogrowth (hay Autoshrink) thì chúng ta có thể sẽ gặp hai vấn đề sau:

- *Performance hit*: ảnh hưởng đáng kể đến khả năng làm việc của SQL Server. Do nó phải thường xuyên kiểm tra xem có đủ khoảng trống cần thiết hay không và nếu không đủ nó sẽ phải mở rộng bằng cách dành thêm khoảng trống từ đĩa cứng và chính quá trình này sẽ làm chậm đi hoạt động của SQL Server.

- *Disk fragmentation (phân mảnh đĩa)*: Việc mở rộng trên cũng sẽ làm cho data không được liên tục mà chứa ở nhiều nơi khác nhau trong đĩa cứng, điều này cũng gây ảnh hưởng lên tốc độ làm việc của SQL Server.

Trong các hệ thống lớn người ta có thể dự đoán trước kích thước của database bằng cách tính toán kích thước của các tables, đây cũng chỉ là kích thước ước đoán (có thể tham khảo thêm "Estimating the size of a database" trong SQL Books Online của bộ cài SQL Server 2000 để biết thêm về cách tính) và sau đó thường xuyên dùng một số câu lệnh SQL (thường dùng các

câu lệnh bắt đầu bằng DBCC) kiểm tra xem có đủ khoảng trống hay không nếu không đủ ta có thể chọn một thời điểm mà SQL server ít bận rộn nhất (như ban đêm hay sau giờ làm việc) để nới rộng data file, như thế sẽ không làm ảnh hưởng tới hiệu năng hoạt động của Server.

Chú ý: giả sử ta dành sẵn 2GB cho datafile, khi dùng Window Explorer để xem, ta sẽ thấy kích thước của file là 2 GB nhưng data thực tế có thể chỉ chiếm vài chục MB.

2.2. Kiểu dữ liệu

Sau khi đã tạo xong CSDL, vấn đề tiếp theo chúng ta bàn đến là sự hiểu biết của người quản trị CSDL về các loại dữ liệu. Ví dụ: bạn phải biết rõ sự khác biệt giữa *char(10), nchar(10) varchar(10), nvarchar(10)*. Loại dữ liệu Char là một loại string có kích thước cố định, nghĩa là trong ví dụ trên, nếu data đưa vào "This is a really long character string" (lớn hơn 10 ký tự) thì SQL Server sẽ tự động cắt phần đuôi và ta chỉ còn "This is a". Tương tự nếu string đưa vào nhỏ hơn 10 thì SQL sẽ thêm khoảng trống vào phía sau cho đủ 10 ký tự. Ngược lại loại varchar sẽ không thêm các khoảng trống phía sau khi string đưa vào ít hơn 10. Còn loại data bắt đầu bằng chữ n chứa dữ liệu dạng unicode.

Một lưu ý khác là trong SQL Server ta có các loại Integer như : *tinyint, smallint, int, bigint*. Trong đó, kích thước từng loại tương ứng là 1, 2, 4, 8 bytes. Nghĩa là loại smallint tương đương với Integer và loại int tương đương với kiểu *Long* trong VB.

2.3. Thiết kế bảng

Khi thiết kế một table nên:

- Có ít nhất một cột thuộc loại ID dùng để xác định một record dễ dàng.
- Chỉ chứa dữ liệu của một thực thể.

Trong ví dụ sau, thông tin về sách và Nhà xuất bản được chứa trong cùng một table.

Books

BookID	Title	Publisher	PubState	PubCity	PubCountry
1	Inside SQL Server 2000	Microsoft Press	CA	Berkely	USA
2	Windows 2000 Server	New Riders	MA	Boston	USA
3	Beginning Visual Basic 6.0	Wrox	CA	Berkely	USA

Ta nên tách ra thành table Books và table Publisher như sau:

Books

BookID	Title	PublisherID
1	Inside SQL Server 2000	P1
2	Windows 2000 Server	P2
3	Beginning Visual Basic 6.0	P3

và Publishers

PublisherID	Publisher	PubState	PubCity	PubCountry
P1	Microsoft Press	CA	Berkely	USA
P2	New Riders	MA	Boston	USA
P3	Wrox	CA	Berkely	USA

- Tránh dùng cột có chứa NULL và nên luôn có giá trị Default cho các cột.
- Tránh lập lại một giá trị hay cột nào đó.

Ví dụ: một cuốn sách có thể được viết bởi hơn một tác giả và như thế ta có thể dùng một trong hai cách sau để chứa data:

Books

BookID	Title	Authors
1	Inside SQL Server 2000	John Brown
2	Windows 2000 Server	Matthew Bortniker, Rick Johnson
3	Beginning Visual Basic 6.0	Peter Wright, James Moon, John Brown

hay

Books

BookID	Title	Author1	Author2	Author3
1	Inside SQL Server 2000	John Brown	Null	Null
2	Windows 2000 Server	Matthew Bortniker	Rick Johnson	Null
3	Beginning Visual Basic 6.0	Peter Wright	James Moon	John Brown

Tuy nhiên việc lập đi lập lại cột Author sẽ tạo nhiều vấn đề sau này. Chẳng hạn như nếu cuốn sách có nhiều hơn 3 tác giả thì chúng ta sẽ gặp nhiều phiền phức. Trong ví dụ này ta nên chặt ra thành 3 table như sau:

Books

BookID	Title
1	Inside SQL Server 2000
2	Windows 2000 Server
3	Beginning Visual Basic 6.0

Authors

AuthID	First Name	Last Name
A1	John	Brown
A2	Matthew	Bortniker
A3	Rick	Johnson
A4	Peter	Wright
A5	James	Moon

AuthorBook

BookID	AuthID
1	A1
2	A2
2	A3
3	A4
3	A5
3	A1

2.4. Quan hệ giữa các bảng

Một vấn đề quan trọng mà chúng ta cần biết đến trong thiết kế CSDL là phải biết rõ quan hệ (Relationship) giữa các table:

- *One-to-One Relationships* (quan hệ một-một): trong mối quan hệ này thì một hàng bên table A không thể liên kết với hơn 1 hàng bên table B và ngược lại.

- *One-to-Many Relationships* (quan hệ một-nhiều): trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B.

- *Many-to-Many Relationships* (quan hệ nhiều-nhiều): trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B và một hàng bên table B cũng có thể liên kết với nhiều hàng bên table A. Như ta thấy trong ví dụ trên, một cuốn sách có thể được viết bởi nhiều tác giả và một tác giả cũng có thể viết nhiều cuốn sách. Do đó mối quan hệ giữa Books và Authors là quan hệ Many-to-Many. Trong trường hợp này

người ta thường dùng một table trung gian để giải quyết vấn đề (chính là table AuthorBook).

Để có một database tương đối hoàn hảo nghĩa là thiết kế sao cho data chứa trong database không thừa không thiếu, chúng ta còn cần biết thêm về các thủ thuật tiêu chuẩn hoá (Normalization). Tuy nhiên, trong phạm vi báo cáo thực tập chuyên ngành này, sinh viên cũng chưa tìm hiểu kỹ về lĩnh vực này nên chúng ta chưa bàn luận tới ở đây.

Tóm lại, trong chương này, chúng ta đã tìm hiểu về cấu trúc của một SQL Server database và một số vấn đề cần biết khi thiết kế cơ sở dữ liệu.

TÀI LIỆU THAM KHẢO

1. **Vụ Giáo dục chuyên nghiệp**, ĐỀ CƯƠNG MÔN HỌC, 2003.
2. **Lê Tiến Vương,** NHẬP MÔN CƠ SỞ DỮ LIỆU QUAN HỆ, NXB Thống kê, 2000.
3. **Nguyễn Kim Anh**, NGUYÊN LÝ CỦA CÁC HỆ CƠ SỞ DỮ LIỆU, NXB ĐHQG Hà Nội.
4. **Claude Delobel, Michel Adiba,** BASES DE DONNÉES ET SYSTÈMES RELATIONNELS, DUNOD, Paris, 1982.
5. **Codd, E.F** (1970), A RELATIONAL MODEL FOR LARGE SHARED DATA BANKS, Comm, ACM 13:6, pp. 377-387.
6. **Chen, E.** PF(1984), OPTIMAL COMPUTATION OF TOTAL PROJECTIONS WITH UNIONS OF CHASE JOIN EXPRESSION, ACM SIGMOD Intl.Conf on Management of Data, pp. 149-163.

MỤC LỤC

Phần 2
HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL

Chịu trách nhiệm xuất bản:

Chủ tịch HĐQT kiêm Tổng Giám đốc NGÔ TRẦN ÁI

Phó Tổng Giám đốc kiêm Tổng biên tập VŨ DƯƠNG THỤY

Biên tập lần đầu:

HOÀNG TRỌNG NGHĨA

Biên tập tái bản :

BÙI MINH HIỂN

Trình bày bìa :

TÀO HUYỀN

Sửa bản in:

TUẤN HIỆP

Chế bản :

ANH ĐỨC

Giáo trình CƠ SỞ DỮ LIỆU

Mã số: 6H158T5 - DAI

In 2.000 cuốn, khổ 16 x 24 cm. Tại Công ty in Thái Nguyên. Số in: 1331. Số xuất bản: 21/246-05 CXB. In xong và nộp lưu chiểu tháng 3 năm 2005.



CÔNG TY CỔ PHẦN SÁCH ĐẠI HỌC - DẠY NGHỀ
HEVOBCO
Địa chỉ : 25 Hàn Thuyên, Hà Nội



NGÔI SAO BẠCH KIM
CHẤT LƯỢNG
QUỐC TẾ







Giá: 11.500 đ